MANAGING YOUR HARD DISK

que

Managing Your Hard Disk

Don Berliner

with Chris DeVoney

Que™ Corporation
Carmel, Indiana

Library of Congress Catalog No.: 86-61154
ISBN 0-88022-265-4

90 89 88 87 8 7 6 5 4 3

Interpretation of the printing code: the rightmost double-digit number is the year of the book's printing; the rightmost single-digit number, the number of the book's printing. For example, a printing code of 87-4 shows that the fourth printing of the book occurred in 1987.

Dedication

To Didi and Hillary, my closest and strongest fans

To Nick, one heck of a kid, whose perseverance never wavered

Development Director
David Paul Ewing

Product Director
Chris DeVoney

Editorial Director
David F. Noble, Ph.D.

Managing Editor
Gregory Croy

Editors
Jeannine Freudenberger, M.A.
Lois Sherman
Lloyd J. Short
Betty White

Technical Editors
Dirk Johnson
David Nickolich

Production Foreman
Dennis Sheehan

Production
Kelly Currie
Jennifer Matthews
Joe Ramon
Mae Louise Shinault
Peter Tocco
Lynne Tone

Composed in Garamond and Digital
by Que Corporation

Cover designed by
Listenberger Design Associates

About the Authors

Don Berliner has been involved in the computer business for over 23 years and has been working with microcomputers since 1978. A free-lance writer, Don currently is working for a major Fortune 200 firm, where he provides decision system and other microcomputer applications support for the marketing area. Don has written articles for business publications and a number of computer software and hardware reviews. He is the author of *Want a Job? Get Some Experience; Want Experience? Get a Job!* published by the American Management Association, and the reviser for Symphony Release 1.1 of *Using Symphony*, published by Que Corporation. Don also has started a telephone call-in and talk show "Computers and You" on his local radio station.

Chris DeVoney, an officer of Que Corporation, received his B.S. degree from the University of Gloucester and has been employed in the microcomputer industry since 1975. Mr. DeVoney has written and edited numerous books and articles about microcomputers. He is author of *IBM's Personal Computer* (first and second editions), *MS-DOS User's Guide*, and *Using PC-DOS*, all published by Que Corporation.

Table of Contents

Acknowledgments

The authors wish to thank William Reekes, technical sales manager at Seagate Technology, Scotts Valley, California, for his assistance in providing useful technical information.

Trademark Acknowledgments

Que Corporation has made every effort to supply trademark information about company names, products, and services mentioned in this book. Trademarks indicated below were derived from various sources. Que Corporation cannot attest to the accuracy of this information.

1-2-3, Lotus, and Symphony are registered trademarks of Lotus Development Corporation.

1 DIR is a trademark of Bourbaki Incorporated.

BackTrack is a trademark of Tallgrass Technologies.

Bernoulli Box and Iomega are registered trademarks of Iomega Corporation.

Brief and UnderWare are trademarks of UnderWare, Inc.

CHART-MASTER is a registered trademark and DataGrabber is a trademark of Decision Resources, a division of Ashton-Tate Company.

COMPAQ and COMPAQ DeskPro are registered trademarks of COMPAQ Computer Corporation.

dBASE III Plus, Framework, and Ashton-Tate are registered trademarks and Framework II and FRED are trademarks of Ashton-Tate Company.

DESQview is a trademark of Quarterdeck Office Systems.

Disk Optimizer is a trademark of SoftLogic Solutions.

DSBackup is a trademark of Design Software Incorporated.

FastBack is a registered trademark of Fifth Generation Systems, Inc.

FilePath is a copyright of SDA Associates.

Flash is a trademark of Software Masters.

GEM is a trademark of Digital Research Inc.

Hercules Graphics Card is a trademark of Hercules Computer Technology.

IBM and TopView are registered trademarks and PC XT and Personal Computer AT are trademarks of International Business Machines Corporation.

Irwin 310 BACKUP Tape Subsystem is a trademark of Irwin Magnetics.

Lightning is a trademark of Personal Computer Support Group.

Mace+ Utilities is a trademark of Paul Mace, Inc.

Macintosh is a trademark of McIntosh Laboratory, Inc., licensed to Apple Computer, Inc., and is used with its express permission.

Mountain is a registered trademark of Mountain Computer Inc.

Microsoft and MS-DOS are registered trademarks and Microsoft Word and Microsoft Project are trademarks of Microsoft Corporation.

MicroSpell is a trademark of Trigram Systems.

Mirror is a registered trademark of Corvus Systems, Inc.

MultiMate Advantage is a trademark of Multimate International, an Ashton-Tate Company.

NewWord is a registered trademark of Newstar Software, Inc.

Norton Utilities, The Norton Commander, and Unerase are trademarks of Peter Norton Computing.

Paradox is a trademark of Ansa Corporation.

PathMinder is a trademark of Westlake Data Corporation.

PCTools is a trademark of Central Point Software.

PC-Sweep is a trademark of Koch Software Industries.

PFS:FILE is a registered trademark of Software Publishing Corporation.

Plus is a trademark of Plus.

ProKey is a trademark of RoseSoft, Inc.

Q&A is a trademark of Symantic Corporation.

Quadram is a registered trademark of Quadram.

R:BASE is a registered trademark of Microrim, Inc.

Ready! is a trademark of Living Video Text, Inc.

SideKick is a registered trademark and Reflex and Superkey are trademarks of Borland International, Inc.

SixPack is a trademark of AST Research, Inc.

Sperry IT is a trademark of Unisys.

SuperCalc3 and SuperCalc4 are registered trademarks of Computer Associates International, Inc.

TopView is a trademark of International Business Machines Corporation.

UNIX is a trademark of AT&T.

Videotrax is a trademark of Alpha Micro.

WordPerfect is a registered trademark of WordPerfect Corporation.

WordStar, WordStar 2000, and MicroPro are registered trademarks of MicroPro International Corporation.

XTREE is a trademark of Executive Systems Incorporated.

Introduction

Managing Your Hard Disk is for any potential or current user of a hard disk computer system. The book is written for readers who are purchasing their first hard disk, adding an additional hard disk, learning to use a hard disk, or wanting to get from their hard disk the greatest efficiency with the least effort.

The book emphasizes the two *e*'s: ease of use and efficiency. The techniques and advice in this book are based on the authors' combined thirty years of experience in the computer field.

If you are contemplating purchasing a hard disk, you need this book. It will help you to identify the important issues and factors and ignore meaningless statistics. After reading this book, you will be armed with the information necessary to make a knowledgeable and cost-effective decision. The book also outlines the steps needed to install your hard disk successfully.

If you have purchased a hard disk recently, you need this book to learn the best way to set up your hard disk. The information and techniques unlock the power of the DOS startup commands, which help you use your hard disk more quickly and more easily.

If you have had your hard disk for some time, you can use the information in this book to improve performance. For example, the book provides insight on the best way to organize your files. The book also explains how forethought and planning can produce better results from your hard disk.

We do not believe that DOS provides every answer for hard disk management. DOS is only a starting point. We are firm believers that you must build a "tool chest" of outside programs to manage your hard disk. These outside programs fill the gaps left by DOS. The programs increase hard disk efficiency and make operating the computer system so intuitive that you perform most common and some uncommon operations with little conscious effort. Some of the tools discussed in this book are commercially available; some tools are free. The quality of your use of the hard disk will depend heavily on the quality of your tool chest.

What You Should Have

Starting with Chapter 4, to get the most out of this book, you need the following:

1. Your computer system
2. A hard disk drive (installed or not yet installed)
3. The proper version of MS-DOS™ or PC DOS for your computer (Version 3.1 or 3.2 preferred)
4. A text editor or word processor capable of producing ASCII text (usually through a programmer or nondocument mode)

Starting with Chapter 5, the information is based on the assumption that your hard disk is installed and working. The necessary directions are given in Chapter 4.

What Versions of DOS Are Covered

The information in *Managing Your Hard Disk* is oriented strongly toward MS-DOS and PC DOS Version 3, with Version 3.2 preferred. Most examples in this book were produced using either DOS V3.1 or V3.2.

The book does cover DOS V2.1, but we strongly recommend that you upgrade to a later version. Certain features of DOS V3 offer greater ease and better performance to a hard disk user. Moreover, certain problems with DOS V2.1 cause difficulties for a hard disk. The current version (V3.2) offers more commands and is more reliable than Versions 3.0 and 3.1.

What This Book Covers

This book is divided into four major sections. Each section concentrates on a different area of hard disk management. The information ranges from tips for purchasing a hard disk through installation and basic use to the choice of software.

Part I introduces you to hard disks and takes you through the steps of choosing, purchasing, and installing a hard disk. In this part you learn how to get your hard disk system up and running efficiently.

Chapter 1 examines the major differences between floppy diskettes and hard disks. The discussion touches on both the advantages and hazards of working with a hard disk. Chapter 2 explores the relationship between DOS and a hard disk. In this chapter, you learn what DOS does with a "raw" hard disk to prepare and organize it for receiving data. The chapter includes a discussion of DOS file names and default conventions.

Chapter 3 takes you through the points you need to consider in choosing a hard disk. This chapter includes a discussion of how your present equipment should interact with a hard disk and an interpretation of the common hard disk statistics. The chapter also introduces the topic of backup; that is, making duplicate copies of your work to protect against inadvertent data loss.

The installation of a hard disk is covered in Chapter 4. This chapter also describes how to set up your system *after* the disk is installed so that it is ready to receive information. You learn how to create and use the two key files (AUTOEXEC.BAT and CONFIG.SYS). These are the files that leap into action whenever you *boot up*, or start, your computer.

Part II provides an overview of the basic information you need before you begin working with your hard disk. Although this section is by no means a full course in DOS, the text reviews the specific parts of DOS that are essential for using a hard disk.

Chapter 5 explores the usefulness of hierarchical directories. You learn some basic organizational principles and the advantages of using subdirectories. The topics include creating, changing, and deleting subdirectories on your hard disk. Chapter 6 reviews some DOS basics that are helpful when working with a hard disk. Specifically, the following DOS commands are introduced: COPY, VERIFY, COMP, DEL, ERASE, TREE, and PATH. You learn to use these commands to increase the efficiency of your hard disk.

The DOS batch file capabilities are discussed in the next two chapters. Chapter 7 uses simple examples to introduce batch files and the batch file commands and parameters. The examples and the discussion provide hints for ways to use batch files to make your work easier and more effective. Chapter 8 deals with advanced batch file techniques, including tips for greater efficiency and warnings of mistakes to avoid. You are given techniques for running batch files from other batch files; suggestions for additions to your AUTOEXEC.BAT file; and sample menu, help, and interactive batch files.

Chapter 9 delves further into the subject of hierarchical directories and tells how to organize the hard disk to accommodate the different ways that programs use files. You are taken through the steps necessary to create different levels of subdirectories. The chapter also discusses uses of the DOS ASSIGN, SUBST, and PATH commands.

Part III is a trouble-shooting section. You explore the many DOS commands available to hard disk users and look at some separate utility programs that assist in working on a hard disk. Chapters 10 and 11 cover DOS commands in file, directory, and disk operations, including the early detection and correction of certain hard disk problems. In Chapter 12, you see how to fine-tune the performance of the hard disk by using RAM, disk cachers, and software programs. You learn many ways to use your disk storage more efficiently.

Part IV is devoted to ways to improve hard disk management. You are introduced to techniques and programs for organizing your disk and performing backups. This section also provides brief reviews of specific software environments and programs.

Chapter 13 describes a sampling of programs in the category of hard disk management software. Commercial, public-domain, and shareware programs are discussed. These programs provide various combinations of menu systems, hard disk file and directory management, and on-line help screens. Some specific programs included are Fixed Disk Organizer™, XTREE, The Norton Commander™, PCTools™, and G-WHIZ.

Chapter 14 extends the preceding discussion to include four alternative operating environments: GEM™ Desktop, Microsoft™ Windows, TopView™, and DESQview™. The programs are described with ideas for good ways to use each program on your hard disk. You learn how your productivity can be effectively increased by these programs.

Detailed information regarding the subject of backup is covered in Chapters 15 and 16. The DOS commands BACKUP and RESTORE; the new command XCOPY; and software alternatives FastBack™, BackTrack™, and DSBackup™ are discussed. Also covered are examples of some hardware alternatives: tape drives, the Bernoulli Box™, and VCRs.

Chapter 17 briefly discusses some of the more popular software packages and tells how to set up and use them most effectively on a hard disk. Although only 15 programs are covered, they are representative of the different kinds of programs available. In the discussion, the chapter points out principles you can use to work with *any* software package on a hard disk.

What This Book Does Not Cover

Some assumptions have been made about you, the reader of this book. The book does cover many DOS commands concerning hard disk use, and some DOS commands not related to a hard disk. *Managing Your Hard Disk* is not intended as a tutorial on DOS or as a complete DOS users' guide. If you are a newcomer to DOS, you will benefit by purchasing another book on DOS, such as Chris DeVoney's *Using PC DOS* published by Que Corporation, and by using *Managing Your Hard Disk* with your DOS users' guide.

The book does not cover the more technical aspects of hard disk use. Although an overview of technical issues is given, subjects of interest to system integrators or programmers are not included. For example, technical subjects not covered are

The details of hard disk adapters
The writing of device drivers
Physical and logical interfaces for disk drives
The use of DEBUG to alter DOS

These subjects are suited to system programmers, not to most hard disk users.

Products and Improvements

Throughout the book, programs not provided with DOS are discussed. Third-party hardware products, such as disk drives and backup tape equipment, also are discussed. In an industry as dynamic as personal computers, the only constant is change. Programs are revised or replaced frequently, and equipment is upgraded regularly. Undoubtedly, by the time you read this book, improvements in the products mentioned will have been made. New products very likely may overtake established products.

Therefore, the specific products cited should be viewed as representative samples, which are based on the product version available at the time of this writing. Consider newer products or versions as added benefits, not as added confusion. Many people are satisfied with earlier version of programs or equipment. Technologically "keeping up with the Joneses" is not always necessary. Newer versions seldom make obsolete what you already

know about a program or product. For example, most improvements do not change commands, but instead the new versions increase functionality with additional commands.

Conventions Used in This Book

You should become familiar with the shorthand notations used in this book. You need not memorize the conventions now. All are discussed again in the appropriate sections of this book.

The most common notation is for issuing commands and running programs. You will often see lines that look like

*dc:pathc***PROGRAM_NAME** *optional_information /optional_switches*

Two different type faces are used to distinguish the parts of a command you must give and the parts of a command you give only when necessary. Items in **boldface** are mandatory. You must always give this part of a command. In this example, to run the program called PROGRAM_NAME, you must type this name. Items in *italic* are optional. The items are given only when needed. In the preceding example, the items before and after **PROGRAM_NAME** are optional. Words in UPPERCASE letters must be typed letter for letter. Items in lowercase are variables. You substitute the appropriate disk drive, file name, or other information for the lowercase name.

In the example, **PROGRAM_NAME** (boldface and uppercase) must be typed as it appears. *dc:pathc*\, *optional_information*, */optional_switches* (italic and lowercase) are optional variables. You substitute the appropriate disk drive and path name for *dc:pathc*\. You substitute a file name or other item for *optional_information*. You use the appropriate character or characters for */optional_switches*. You simply drop the parts that are not needed. This topic is explained in more detail in Chapter 2.

You can type commands in uppercase or lowercase letters. The case usually does not matter. I usually use uppercase letters in the examples, but I could have used lowercase or used both uppercase and lowercase letters instead.

In a few commands the case of the letters is significant and does make a difference. When the case is important, I specifically mention the difference when discussing the command. Otherwise, when you enter commands, you can use safely either uppercase or lowercase letters or both.

Part **I**

Selection and Setup

Part I of this book is for both the newcomer and the experienced user. If you are just starting with your computer, this part gives you the necessary facts about a hard disk, including how to select the "best" hard disk for your system. If you already have a hard disk, the text lays the foundation for the techniques recommended later in the book. The experienced user can consider Part I a hard-disk refresher course. You may find the information interesting, and you may learn some facts you did not know before. On the other hand, you may find that skimming the chapters is most productive for you.

The material covered in this part includes a discussion of the background of the hard disk and explains hard disk terminology, functions, features, and the differences between hard disks and floppy disk drives. The practical, hands-on information you find throughout this text begins in this section, including the physical and logical installation of a hard disk and the starting setup.

Part I has four chapters. The first chapter is the hard disk primer. It introduces the parts of the hard disk and also exposes some of the potential joys and troubles of owning a hard disk system. Chapter 2 explores how DOS works with and controls the hard disk. The most important concepts concern file and disk drive names and the information DOS maintains about each file.

Chapter 3 presents information you need when you are shopping for a hard disk. You learn that some features are vital and some statistics are worthless. Although the chapter is written for those searching for a hard disk, the text introduces some additional computer terms and concepts used later in the book.

Chapter 4 is the installation chapter and explains both physical and logical installation. This chapter also covers the automatic start-up functions of DOS (the AUTOEXEC.BAT file) and suggests ways to add or alter DOS functions through a configuration file.

You may want to follow this suggested "road map" for this part. If you're already familiar with hard disks, you might skim Chapter 1 and go quickly to Chapter 2. If you are familiar with how DOS works but are in the market for a hard disk, skim the first two chapters and start reading with Chapter 3. If you are satisfied with your current hard disk setup, the essential information you should know is the latter part of Chapter 4 in the material on the AUTOEXEC.BAT file and the CONFIG.SYS, configuration, file. If you are familiar with DOS and comfortable with your computer setup, proceed to Part II.

Do not feel obligated to memorize every fact presented in this part. Although later chapters build on the information presented in Part I, the purpose of this part is to acquaint and inform. To do both, some mildly to moderately technical information is presented. This information easily can overwhelm the newcomer.

You need not know everything about DOS to use a computer. You need not know all aspects about the parts of a hard disk to choose a hard disk. You need not know all the inner workings of a computer to set up a hard disk.

For most people, the information presented in this part can be absorbed in one reading. My advice is simply to grasp the key terms and most important concepts as you read each chapter, then move on. You can return later to further your understanding. As you start, the *what* you know is more important than the *why* or *how*. As your knowledge grows, the reason (*why*) and the technique or technology (*how*) fall into place.

Chapter 1

An Introduction to the Hard Disk

To some microcomputer users, a hard disk is heaven. To others, a hard disk is a matter of fact, accepted as the normal state of affairs. Most users who have hard disks don't want to do without them. If you have never used a hard disk, you can easily become addicted to its convenience and speed. If you have just started using one, you will soon have trouble imagining computing without one. If you have always used a hard disk, occasional problems remind you how convenient the hard disk is.

In contrast to a floppy diskette, a hard disk doesn't have to be placed into the computer each time you need to access data; the hard disk is already there (either hidden inside the computer or as a separate external box). The only indication that the disk is operating is a flickering red indicator light and the noise the recording heads make.

If you have ever played the "floppy shuffle" trying to find the right floppy diskette or have had to switch program or data diskettes, you can easily understand a user's love affair with the hard disk. The hard disk offers storage capacity equivalent to a load of floppy diskettes. A 10-megabyte hard disk (seemingly the bare minimum nowadays) equals about 28 double-sided floppies. Storing and retrieving data and running programs with a hard disk is tremendously convenient. The only time some people use the floppy disk drive is to load new

programs on the hard disk or to back up data. A hard disk drive is also faster than floppy disks. Information is retrieved and stored three to ten times faster on a hard disk than on a floppy disk.

With prices for hard disks, like prices for most electronic components, dropping rapidly, including or adding a hard disk to your computer becomes easier and easier.

This chapter presents a brief look at the hard disk. The text defines a hard disk and explains the types of hard disks available. You will learn why hard disks are "superior" to floppy disks and what problems can occur with hard disks.

Hard Disk Basics

This section introduces some of the concepts of hard disk storage. Use this section as a refresher on disk storage terms and ideas.

Everyone using a personal computer should already be familiar with the concept of a computer disk. The disk is a cross between a phonograph record, a digital stereo disk (often called a CD for compact disk), and a tape recorder. The disk gets its name from its circular shape, like a phonograph record or compact stereo disk. Like the media of the tape recorder, the metal oxide coating of the disk is selectively magnetized to hold information. Like a tape recorder, the disk drive has one or more recording heads. Unlike the tape recorder, the disk drive usually has one recording head per side of the disk. Like a phonograph, the heads are mobile.

The phonograph and tape recorder play and record a range of information, an analog recording. The disk is more similar to a compact stereo disk; both reduce information to the simple on-and-off digital state. The CD uses a laser light to retrieve audio information; the computer disk drive recording heads read the state of magnetic particles on the disk to retrieve information.

Floppy diskettes get their name from their flexible media and envelope. The diskette is made of a plastic material coated with metal oxide. The envelope is made of a flexible PCB plastic. When you shake the diskette, it flops.

Hard disks, similarly, are named for their rigid platters. Usually made of steel, the platters are coated (or plated) with a denser metal oxide than the type used on a floppy diskette. The platters (typically, more than one platter is used) are mounted on a metal spindle.

Hard disks generally come in two "flavors": removable and nonremovable. *Removable hard disks* are simply units in which the

hard disk platters can be changed, similar to the way a floppy disk is changed. The removable media is enclosed in a plastic case. Because the recording heads are in the disk drive, the housing has access holes for the recording heads. The removable hard disk drive holds the recording heads, the motor to spin the platters, and the necessary electronics to control the parts of the disk drive and communicate to the computer.

To prepare a removable hard disk for daily use, you open a door or hatch on the disk drive, mount the platter or platters, and close the hatch. The air inside the drive is purged; the recording heads move into place; the platters are "brought up to speed" (spun at the proper rate); and the drive is ready for use.

Nonremovable disk drives, the more common type of hard disk drive, are "all in one" assemblies. The platters, recording heads, motor, and associated electronics are contained in one unit. The popular name for this type of unit is a *Winchester* disk, the code name for this 1973 IBM® invention. Winchester technology was developed to increase the speed and storage size of hard disks. The first Winchesters were 3340, cost $8,600, and held 140 megabytes. The disk drives were used exclusively on mainframe computers because of their high price.

With the drastic decline in price during the last 15 years, the Winchester has become the most popular type of disk drive for personal computers. In fact, IBM and many other companies have adopted the Winchester for use on personal computers.

You may note that many computerists use the term *hard disk*; IBM uses the term *fixed disk*; and others use the terms *Winchester*, *removable*, and *nonremovable* disk. These terms are almost synonyms, but not quite. A hard disk is any disk drive that uses hard platters as the media. A *Winchester disk drive* is any disk drive with the platters and recording heads sealed in one package. IBM uses the term *fixed* because the assembly holding the platters and recording heads cannot be removed from the Winchester disk drive. Hence, *nonremovable* and *fixed* are synonyms. A fixed disk drive is a nonremovable Winchester disk drive.

The most popular disk drive on the market is the nonremovable Winchester disk drive. Because this single type of hard disk drive dominates the personal computer scene, the entire personal computer community has fostered a sloppy interchanging of the terms *hard disk*, *Winchester disk*, and *fixed disk*. In this book, the term *hard disk* generally refers to all types of disk drives. When the distinction is important, a disk drive is referred to by its exact name.

Not all Winchester disk drives are nonremovable. A hybrid removable Winchester hard disk is available for both large and small computers. The sealed assembly holding the platters and recording heads is called a *data pack*. This data pack is mounted on a disk drive, which holds a motor and the connecting electronics. Most of these types of removable Winchester disks hold 50 to 800 million bytes of data and are not cost justified for use with a microcomputer. Syquest, however, makes a true removable Winchester unit for microcomputers.

Hard disks for personal computers usually have platters with diameters of 3 1/2, 5 1/4, 8, or 14 inches. The most common is the 5 1/2-inch platter, followed by the 3 1/2-inch size. The units are about the same size as the equivalent floppy disk drives. Like floppy disks, the disk drives come in full-height (about 3 inches tall) and half-height (about 1 1/2 inches tall) models. Functionally, no difference exists between full-height and half-height hard disks.

The usual capacity of 3 1/2-inch hard disks is between 10 and 30 million characters of information (millions of bytes or *megabytes*). The 5 1/4-inch disk drives normally have capacities of 10 to 40M. (M is the abbreviation for megabyte.) Most of these disk drives are in the $300 to $2,000 range, but you can go far beyond these figures. For example, you can buy a full-height Maxtor 5 1/4-inch Winchester disk drive that holds 384 megabytes. However, that disk drive sells for about $6,000.

The Speed of the Hard Disk over the Floppy

The hard disk drive has four main parts: the platters, the recording heads, the drive motor, and the drive electronics (see fig. 1.1). The hard platters are mounted vertically on a spindle. A drive motor may be connected directly to the spindle (direct-drive) or may be connected by a pulley-and-belt system. Either way, the platters spin at a rate of 3600 rpm. A floppy disk normally spins at 300 rpm. Because of the faster spin rate, the recording heads cover much more ground (disk surface) on a hard disk drive than on a floppy disk drive during the same time period.

A recording head is attached to an arm, one arm on each side of the platter. These arms move as a single unit, as does a floppy disk drive's, but in smaller, more precise increments. The recording heads

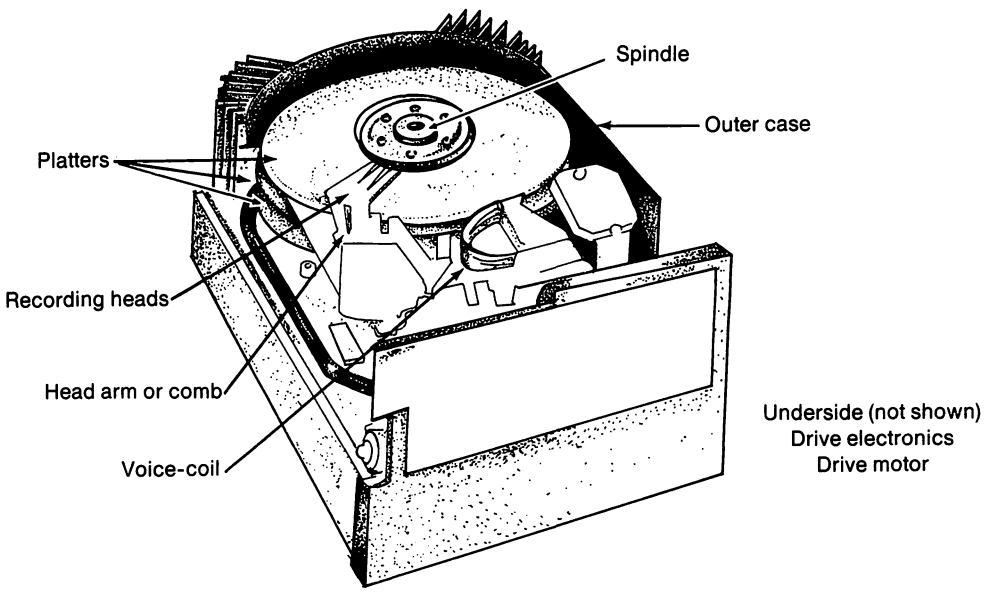


Fig. 1.1. *The anatomy of a fixed disk drive.*

of a hard disk drive are smaller than a floppy disk drive's. With more controlled movement and smaller recording heads, the hard disk platter surfaces can receive more information.

The mobile arms holding the recording heads move faster than the recording heads on a floppy disk. Because of the faster speed of the platters and recording heads, information is stored and retrieved faster on a hard disk than on a floppy disk.

Two recent innovations are increasing the storage capacity and speed of hard disks. Newer recording heads use a different construction and different mountings, called *Whitney technology* (see fig. 1.2). The lighter heads can be moved faster and record even more information on less disk space. The media also has changed from a simple metal-oxide coating to a thinner but denser layer of plated oxide material called *thin-film* media (see fig. 1.3). This media can hold more information and is more resistant to impact, a factor discussed later in this chapter.

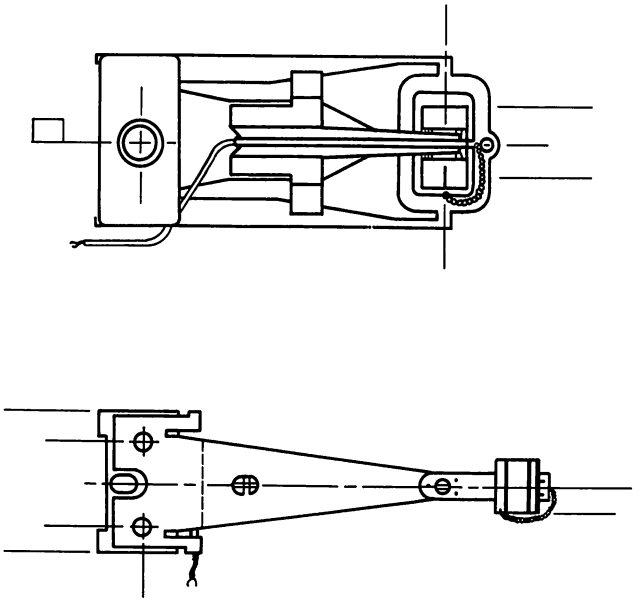


Fig. 1.2. *An older style Winchester recording head (above) and the newer Whitney head (below).*

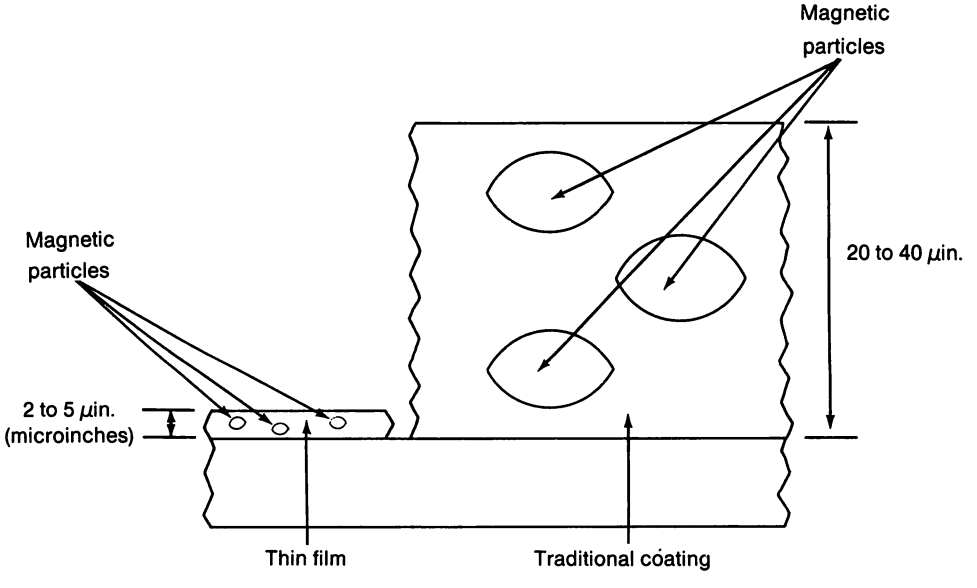


Fig. 1.3. *A comparison of traditional platter coating (right) and thin-film coating (note the difference in thickness).*

The electronics are more sophisticated in a hard disk drive than in a floppy disk drive. Generally, the electronics in the disk drive receive and interpret high-level commands from the computer. The electronics position the recording heads, cause the information to be written or read, and transmit the information between the drive and computer at a high rate of speed. Most hard disks incorporate routines that automatically verify recorded information and correctly recover information which has minor flaws. (The technical term is ECC or error checking and correction.)

Overall, the hard disk is a fast yet complex peripheral. Capable of quickly storing and retrieving millions of bytes, the hard disk has moved from the world of mainframes to the personal computer.

The Hazards of Hard Disks

Many people are disappointed to discover the hard disk drive's vulnerability. Perhaps they have been lulled by the sophistication of microcomputer technology, but the fact is that the hard disk *will* fail eventually. The tough part is predicting just when that failure will happen. Disk failure can occur at any time during a broad time span (usually between 8,000 and 20,000 hours of use).

This statement is not to say that hard disks are unreliable. They are carefully engineered, robust electronic products. But anything made by human beings will fail, and hard disks certainly are no exception.

All disk drives can have problems, from garbled recordings to mechanical failures. Just as some problems are unique to floppy disks, some problems are unique only to hard disks, and some problems are unique to certain types of hard disks. The most common problems are mechanical failures, electronic failures, and data failures.

Mechanical Failures

Mechanical failures usually come in two flavors: drive-motor or spindle-related failures and disk head crash. Neither is pleasant, but the latter is more fatal to information.

Drive-motor and spindle-related problems occur when the platters do not spin at the proper rate or do not spin at all. The most frequent causes are a failure in the ball bearings in the spindle or motor, a loose or broken belt connecting the spindle to the motor, or a burned-out motor.

Of the three problems, the loose or broken belt is the easiest to fix. The bad news is that most contemporary hard disks use direct-drive motors and have no belt to fix. For these, the motor or spindle bearings are frequent mechanical problems. Fortunately, this type of failure usually announces itself long before the problem is fatal. Occasionally, you'll hear an unusual medium- to high-pitched hum or whine coming from the disk drive. That whine is a definite cue to back up your hard disk and check it in for repair as soon as you can.

Head crash is caused when the recording heads literally nose-dive into the platter's surface. The crash is caused by contamination or physical shock.

Unlike the heads in floppy disk drives, hard disk heads ride a cushion of air just thousandths of an inch off the platter surface. The spinning disk creates the air pressure to lift and float the heads. If the heads encounter any contamination (a hair, cigarette smoke, or even a fingerprint), they bounce over the contamination and crash into the platters, causing the infamous head crash (see fig. 1.4). The recording heads, the media, and data on the media can be damaged. The result is expensive: in money, for the repair or replacement of the disk drive; in time, for the replacement of the data.

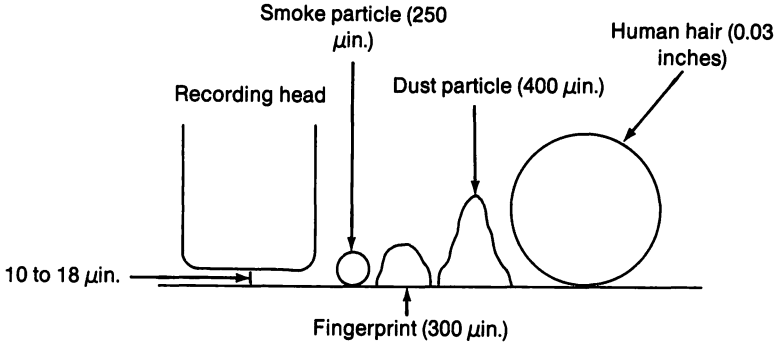


Fig. 1.4. *Sizes of smoke, fingerprint, dust, and hair particles compared to the gap between the platter surface and the recording head.*

When a removable hard disk is placed into the disk drive, the air is forced through a fine filter. This purging removes most airborne contaminants. However, the purging cannot always remove the dangers of smoke, ash, liquids, or the touch of a human hand, any one of which can ruin the disk drive. For this reason, removable hard disk platters are always left in their plastic housing to protect the

platters. Because this housing is not airtight, many data processing shops still post "no smoking" signs near disk storage sites for fear of smoke entering the hard disk cartridges.

Only non-Winchester disks have this problem. Winchesters are hardier. A major advantage to the Winchester disk is the sealed environment, which prevents contamination from getting to the platters. A fine filter allows only air to enter the chamber in order to equalize pressure. Nothing short of dunking the Winchester disk drive in liquid allows any other substance in. This factor is a major reason for the Winchester's popularity in the personal computer market.

All hard disks, however, suffer from the danger of physical shock. If the unit receives a moderate blow when running or is severely jarred when resting, the heads can smack the surface. For this reason, you should not drop or bump a hard disk indiscriminately or strongly jar the table or desk that holds the disk. When at rest, hard disks are more tolerant of shock but not too much more. Most removable hard disks are rated to a shock of 5 to 7 Gs when operating and 10 to 15 Gs at rest. (A G is the force of Earth's gravity.)

Winchester disks can take more punishment than removable hard disks: 10 Gs when in use and 20 Gs when at rest. Newer Winchesters are even more robust. The newer Winchesters use lighter recording heads, which hit the disk with less force and so cause less damage. The plating process of thin-film media makes the platter more resistant to damage. These drives are rated to 15 to 20 Gs at work and up to 30 Gs at rest. If you are in doubt about your Winchester, however, stick to the 20 G and 10 G restrictions for safety's sake.

Electronic and Data Failures

Electronic failures can cause the hard disk to stop operating. Data failure occurs when information cannot successfully be stored and retrieved on the hard disk. Unfortunately, the first type of failure can cause the other.

When any part of the disk's circuitry malfunctions, an electronic failure occurs. Usually, the disk drive acts erratically before failing completely. A common initial symptom is DOS reporting that it cannot read or write a sector. These types of errors usually signal that a failure is imminent.

For this discussion, rather than isolating the failure to a part of the disk drive, the more important aspect is whether the failure

damages the data on your disk. No easy method is available to distinguish between the kind of electronic failure that destroys information and the failures that simply make the disk drive unusable until repaired.

Interestingly, the most common data destruction does not result from head crashes or electronic failures. The greatest destruction comes from the computer user who inadvertently gives the wrong command. Although control over this problem is in your hands, do not be overconfident. This cause of data failure is difficult to prevent completely.

Because hard disk failure is unpredictable, frequent backup of the information is essential. Backing up information is so important that the subject is mentioned frequently throughout this book.

In most cases, you cannot remove hard disk platters (except for the less common removable hard disk and the hard-disklike cartridge media). Therefore, you must transfer information to separate media for backup. Many personal computer users still rely on diskettes for backups; others prefer tape backup units. Establishing a regular habit of backing up your essential information is important. See Chapter 3 for an initial discussion of issues and questions relating to backup; refer to Chapters 15 and 16 for specifics about the different kinds of backups possible.

Summary

This chapter covers some of the basics for hard disks. You have learned the following major points:

- Hard disks get their name from the rigid platters used in the disk drive.
- Most hard disks fall into two categories: the type with fixed media and the type with removable media.
- The hard disk used most frequently with a personal computer is a Winchester disk, also called a fixed disk by IBM.
- All hard disks fail eventually. Although the typical operating time before a failure is between 8,000 and 20,000 hours, the exact time of the failure is unpredictable. The failure may harm data on the hard disk.
- Backing up the data on the hard disk is essential.

The next chapter discusses DOS, the vital software that makes your computer work for you. The chapter explains what DOS is, the parts of DOS, and how DOS controls the hard disk. If you are already familiar with DOS, skip to Chapter 3, which discusses the factors you should consider when buying a hard disk.

Chapter 2

DOS and the Hard Disk

DOS is the acronym for *D*isk *O*perating *S*ystem, a set of programs that gives you and your applications programs control of the computer's resources.

In "olden days," the operating system (OS) provided a standardized way to control the hardware resources of the computer, including the then modern punch card and magnetic tape storage devices. With the advent of disk storage, the name *Disk Operating System* was given to operating systems that could also control disk drives. Today almost all operating systems are disk operating systems, and the two terms are interchangeable.

This chapter presents an overview of the disk operating system. The text explains the essential parts of DOS, tells how DOS manages disk storage, and explains how to use file names and wild cards. This chapter also explains some key terms and concepts of DOS. If you are already familiar with DOS, you can skip this chapter and proceed to the next chapter, which is about selecting a hard disk.

Comparing PC DOS with MS-DOS

In the world of IBM personal computers and compatibles, you often hear the terms PC DOS and MS-DOS®. PC DOS and MS-DOS are disk

operating systems that are almost identical. To understand the difference between them, you need to know the origins of both.

MS-DOS is the disk operating system offered by Microsoft Corporation®. Microsoft bought the rights to 86-DOS, originated by Tim Patterson of Seattle Computer Products (now of Falcon Technologies), and named it MS-DOS. Since then, Microsoft has upgraded MS-DOS several times and has licensed the software to computer manufacturers.

Like any disk operating system, MS-DOS has two portions: a machine-specific (or machine-dependent) part and a machine-independent part. The machine-independent portion is the same for every computer. The machine-specific portion must be tailored for the specific peripherals used—the different types of video displays, keyboards, and disk drives.

Manufacturers are responsible for tailoring MS-DOS to their computer and may rename their versions of MS-DOS if they wish to do so. Most manufacturers, however, have kept the name MS-DOS rather than introduce names that might confuse newcomers.

IBM is a major exception. PC DOS is IBM's version of MS-DOS for the Personal Computer family. IBM officially calls its DOS the IBM Personal Computer Disk Operating System. Because reporters dislike typing long names, the press abbreviated the term to PC DOS.

PC DOS is basically MS-DOS with some differences. The core of MS-DOS is the core of PC DOS. The machine-specific portions of MS-DOS are tailored by Microsoft and IBM for the IBM Personal Computers and their specific peripherals, just as other versions of MS-DOS are tailored for other computers. Because the machine-independent portions remain the same, the names PC DOS and MS-DOS are almost interchangeable.

Microsoft provides a set of housekeeping programs with MS-DOS. These programs format disks, display lists of files, check the integrity of the disk storage, and perform other similar functions. Like the machine-specific portions of MS-DOS, the housekeeping utilities provided by Microsoft are customized for the specific computer equipment but act the same for almost all compatible computers.

Manufacturers also are free to provide additional programs with their versions of MS-DOS. For example, IBM was the first to provide the hard disk BACKUP and RESTORE utilities. Because many clone manufacturers want to maintain PC-compatibility, BACKUP and RESTORE are now standard utilities provided with any hard disk system.

From this point on in this book, the terms PC DOS and MS-DOS are specifically named only when the differences are meaningful. Otherwise, the term DOS applies to both.

Dividing Up DOS

Just as DOS is divided into machine-dependent and machine-independent sections, it also can be divided functionally and operationally. Functionally DOS consists of four sections: the BIOS, the BDOS, the command processor, and the housekeeping utilities. Operationally, DOS is divided into resident and disk-based programs.

The *BIOS*, *B*asic *I*nput-*O*utput *S*ystem, consists of the heavily customized software routines that control the computer's equipment. They perform such functions as moving the disk drive head one step or moving a character from the keyboard to the video display. For MS-DOS machines, the BIOS is located in a file called BIO.SYS. For IBM computers, the file is called IBMBIO.COM.

The *BDOS*, or *B*asic *D*isk *O*perating *S*ystem, is the control center. The heart of the disk-filing system is located here. BDOS performs high-level functions that regulate the flow of information between the various parts of the computer. The MS-DOS file is called MSDOS.SYS; the PC DOS file holding this program is IBMDOS.COM. Although some customizing of this file is performed by manufacturers, the basic functions of these programs are the same.

The *command processor* has several responsibilities. The command processor contains built-in functions that perform such activities as displaying the names and sizes of files on a disk, displaying the contents of a file, and determining the proper actions when problems occur. The command processor also loads and runs the applications programs; for example, word-processing, spreadsheet, or communications programs. This file is known on all IBM and MS-DOS systems as COMMAND.COM.

The fourth piece consists of the *housekeeping utilities*. These utilities format disks; compare files and disks; back up and restore disk files; and change the workings of the display, the printer, and the communications port. These essential programs are provided with the disk operating system. Because the utilities programs come with DOS, they are considered a part of DOS.

All parts of DOS are provided on a diskette. Because DOS is not built into the computer, the programs must be loaded from the diskette into the computer's memory.

Operationally, DOS can be divided into two parts. The first part is loaded into the computer and stays in memory until the system is shut down. The second part comes and goes as needed. The first part is called *resident*, or *internal*; the second part is considered *disk-based*, or *external*. Although all of DOS is truly disk-based, this operational division is determined by the parts that are loaded and stay and the parts that come and go in memory.

When you start DOS, you load the BIOS, the BDOS, and the command processor. The BIOS comes into memory, relocates itself, and loads the BDOS. Then the BDOS relocates itself and calls the command processor into memory. Except for a small portion of memory that the command processor can free for use by your programs, all three sections reside in the computer and form what users think of as DOS. All together, DOS occupies approximately 40 to 60K of RAM, depending on the particular version of DOS.

Computerists often use the term *booting DOS*, which comes from the phrase "pulling yourself up by the bootstraps." The early computers used toggle switches. With these switches, the operator toggled the necessary instructions to load the first record of the operating system. The first record held the initial software, which gave the computer just enough intelligence to load the remainder of the operating system, hence the terms *bootstrap software*, *boot record*, and *booting*. The terms are still appropriate. When the computer starts, it loads the first record of information from the diskette. This boot record holds the bootstrap program, which brings the rest of DOS into the computer.

Understanding How DOS Divides Disks

DOS controls the disk storage. That task is the major role of a disk operating system. To understand better the terms and concepts used later in this book, you need to know how DOS divides disks and stores information. This section also explains how to calculate the exact capacity of any disk.

DOS does not work with the entire surface of a disk when storing information. To simplify and speed the handling of a disk, DOS divides the disk into smaller pieces. In that way, DOS can move rapidly to the smaller pieces and use them.

DOS divides disks into a series of concentric circles called *tracks*. The number of tracks on your disk depends on the motor that steps

(moves) the recording heads. The smaller the step, the more tracks a disk has (see fig. 2.1).

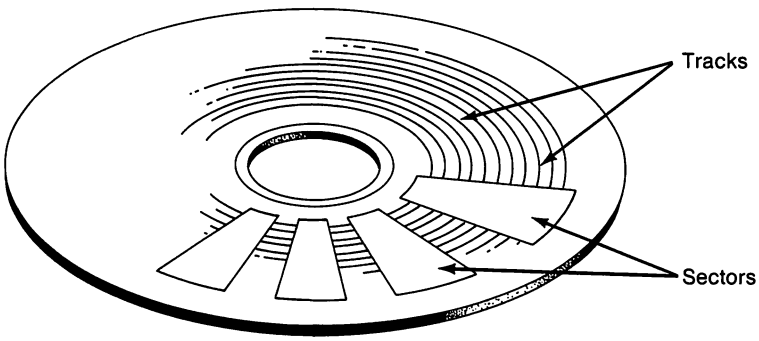


Fig. 2.1. *Hard disk as DOS divides into tracks.*

The term *tracks* is applied more often to floppy diskettes than to hard disks. Hard disk drives use a similar scheme of dividing the surface of each platter, but the tracks align with corresponding tracks on the other surfaces. Each group of aligned tracks is called a *cylinder.*

Think of a double-sided diskette with 40, 70, or 80 tracks on each side. The first track on the top side is directly above the first track on the bottom side, the second track on the top is above the second track on the bottom, and so on, for each of the other 38, 68, or 78 tracks. If you were to draw a vertical three-dimensional figure through both sides of the diskette at any track, you would have a cylinder (see fig. 2.2). Think about two platters of a hard disk stacked on top of each other on the spindle. The tracks for each side of each platter are aligned vertically. If you were to connect any one track through all four surfaces, you would again have a cylinder.

The 20-megabyte fixed disk of the IBM Personal Computer AT™ has 615 cylinders, which means that each surface of the four platters has 615 tracks. The 10-megabyte fixed disk of the PC XT™ has 306 cylinders; each of the four surfaces has 306 tracks. (Both computers use one cylinder for internal purposes. This single cylinder is not available for storing information.)

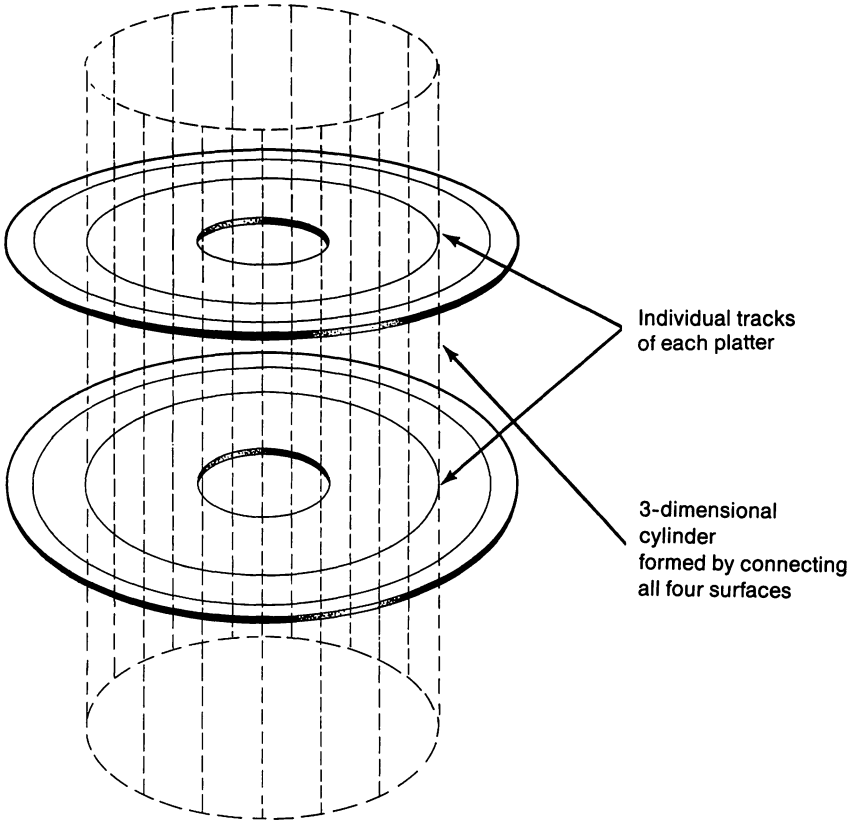


Fig. 2.2 *A typical cylinder for a two-platter hard disk.*

DOS does not use the entire circumference of a track for recording information because the tracks are different lengths. The track closest to the outside edge of the platter would hold more than the track closest to the center of the disk. The overhead of handling these differences by DOS does not justify the benefits of the potentially increased storage. For this reason, DOS slices each track into evenly sized *sectors*. Every sector, regardless of the track, is the same size and holds the same amount of information.

The *bit* (binary digit) is the smallest piece of binary information; the *byte* is the smallest manageable piece of information that holds a single character. The *sector* is the smallest manageable piece of disk storage. Floppy diskettes use 9 or 15 sectors per track; 360K

minifloppy and 720K microfloppy diskettes use 9 sectors; high-capacity 1.2M minifloppy diskettes use 15. The IBM fixed disk drives usually use 17 sectors per track.

When writing information on to a disk, DOS starts with the first side of the first platter, beginning at the first cylinder (track). The first side is number 0. The first platter is number 0. The first cylinder, which is the outermost track, is number 0. The computer world often uses 0, not 1, as the first counting number. The computer world, however, also enjoys breaking rules; the first sector is sector 1.

DOS starts filling sectors in ascending order. When the last sector of that track (9, 15, or 17) is filled, DOS flips to the second side (side 1) and fills track 0, sectors 1 through 9 (15 or 17). On a floppy diskette, which has two sides, DOS then goes back to side 0, steps up to track 1, and proceeds to fill ascending sectors starting with sector 1. The process continues until the disk is full.

The identical process is used with the hard disk. But the hard disk has more platters. DOS fills in sectors 0 through 17 of platter 0, side 0, then flips to side 1. DOS then selects the head for the next track in line—platter 1, side 0; fills in the sectors in the track; switches to side 1 of platter 1; then to platter 2, side 0; and continues to repeat the process. The entire cylinder is filled before DOS steps the recording heads to the next track.

Why does DOS work this way? The answer is speed. Switching between the recording heads takes several microseconds (10^{-4}). Moving the recording heads takes tens to hundreds of milliseconds (10^{-3}). Changing heads is faster than moving the same head on a floppy diskette or a hard disk.

You can find the total "formatted" storage capacity of a disk drive by multiplying the number of cylinders, times the number of platter sides used for recording, times the number of sectors per track, times the number of bytes per sector. To figure the storage, you need to know that most fixed disks use 512 bytes per sector, as do minifloppy diskettes.

The formula for the capacity of a 20-megabyte IBM Personal Computer AT fixed disk is

	615 cylinders
×	4 platter surfaces (2 platters, 2 sides per platter)
×	17 sectors per track
×	512 bytes per sector
=	21,411,840 bytes

If you divide 21,411,840 by 1,048,576 (2^{20} or 1M, a megabyte), the answer is 20.4199M. Because some disk space is needed by DOS and the computer, you actually have slightly more storage than the 20M size implies.

This section has covered how DOS divides a disk and uses those divisions. The following section explains what DOS does to a disk to keep track of your information.

Understanding How DOS Prepares Disks

When you get a new diskette or fixed disk, it is not ready to be used. Some "dummy" information must first be recorded on the disk. This process is called *formatting*.

DOS actually performs two types of formatting: physical (low-level) and logical (high-level). Each type serves a different purpose.

In *physical* formatting, the sides of the diskette or platters of the hard disk are magnetically divided into tracks and sectors. Interleave is established while identification information is marked in each section. Any damaged sectors are marked as such on a special area of the disk. IBM calls this type of formatting *absolute* formatting.

The second type of formatting is the higher-level, or *logical*, formatting. Occasionally, this formatting is called a *relative* format. Logical formatting determines the apparent (logical) structure of the hard disk. When logically formatting the disk, DOS establishes housekeeping sections that keep track of free and in-use areas and of the various files on the hard disk.

You are probably familiar with DOS's FORMAT command, which prepares floppy diskettes and hard disks. You may not know that FORMAT handles the preparation of floppy diskettes and hard disks differently.

FORMAT both physically and logically formats a floppy diskette. But before a hard disk can be logically formatted by FORMAT, the disk must be physically formatted by another program. Disks sold by IBM are physically formatted at the factory. Other disks must be formatted by you or the vendor. Most local dealers do this task for you.

If you buy a hard disk from a mail-order vendor, however, the task falls to you. If you don't have the software to perform the physical formatting, you'll need to purchase it separately. Some, but not all, vendors provide the absolute formatting software.

IBM does not routinely include the software that does physical formatting for hard disks; but an absolute formatting routine is available in the IBM Hardware Maintenance and Service package, commonly known as the Advanced Diagnostics. This package can be purchased for approximately $175 from IBM dealers.

The DOS FORMAT command logically formats a hard disk. In this process, DOS establishes three important areas on the disk. The first area available to DOS holds the *boot record*, mentioned earlier in this chapter. FORMAT places in this area the bootstrap routine DOS uses to load itself. Next, FORMAT sets up two copies of the *file allocation table*, or *FAT*, which keeps track of where files are located. The FAT varies in length, with larger disk drives getting larger FATs. The third area FORMAT establishes is the starting, or *root*, directory, in which DOS keeps six pieces of information about each file:

1. The name
2. The date created or last changed
3. The time created or last changed
4. The attributes (characteristics)
5. The starting cluster entry in the FAT
6. The size in bytes

The boot record, the FAT, and the root directory are essential for DOS's use of the disk.

Program calculations of disk space may not agree with your manual calculations. Programs may state that you have thousands of bytes less disk space than you calculate. The reason for this discrepancy is that all the DOS programs and utilities automatically deduct the disk space used by the boot record, the FAT, and the directory. Because you cannot use these areas to record your own information, DOS does not count these areas in the total disk space.

The Directory and the File Allocation Table

Every file you use or create has one directory entry, 32 bytes long. The starting directory, which spans several sectors, can hold a fixed number of entries. This number varies according to the type of disk you are using (floppy or hard disk) and its capacity (180K, 360K, 720K, 1.2M). Floppy diskettes have maximums of between 62 and 224 file entries; most hard disks, regardless of capacity, can hold 512 file entries in a starting directory.

Although the starting directory cannot hold more than 512 file entries, your hard disk can hold many more files. This limitation is

one reason to use the hierarchical directory system discussed later in the book. You need not understand subdirectories yet. Just remember that the starting directory for your hard disk can hold a limited number of file entries. To exceed this number, you need the hierarchical directory commands.

Some operating systems keep track within the directory of where files are stored on the disk. DOS does not.

With DOS, a file's entry in the directory points to another entry in the file allocation table. The FAT tells DOS which sectors on the disk actually hold the file. In other words, the FAT guides DOS in finding information stored on the disk. The FAT also indicates which sectors are not being used by any file and are available for use.

The FAT is very important to DOS. In fact, the FAT is so important that DOS keeps two copies for every disk. DOS uses the directory entry to go to the correct part of the FAT for information about the file you want. DOS then locates your file on the disk by using the chain of entries in the FAT.

When a file grows, DOS checks the FAT to see what sectors are not being used. When a free sector is found, DOS adds that sector to the chain for that file in the FAT and uses that sector. Thus, the chain of sectors defining a file is linked. When searching for a free sector, DOS uses a first-found, first-used routine. In other words, DOS takes the first free sector found and puts as much of the file as possible in that sector. DOS then looks for the next free sector and places more of the file in the new sector. Depending on the availability of free sectors, a file can be scattered all over the disk.

When discussing the FAT, one additional term is used: cluster. A *cluster* is the smallest unit of disk space that DOS will work with in the FAT. DOS does not actually look for a sector per se but rather for a cluster. The size of a cluster on a hard disk is four, eight, or sixteen sectors, depending on the disk's capacity and the version of DOS you are using. The consequences of scattered files and allocations on a cluster (not sector) basis are discussed in Part III of this book.

Some Special Attributes

The directory also holds other information besides the file's name and location. One convenient feature incorporated into the directory is information about a file's attributes. The attributes tell DOS whether the file needs special handling or protection. The attributes can be any combination of the following:

Read-only	A file that may not be altered or erased.
System	A file used by the operating system. This file is normally hidden from regular DOS operations.
Hidden	Another file hidden from normal DOS operations.
Volume label	Not a disk file at all but an 11-character "electronic label" that helps identify the disk. The volume label shows up when you execute DOS commands like DIR, CHKDSK, or TREE.
Subdirectory	Not a file but a different directory that holds similar information about other files.
Archive	Tells DOS that this file has not been backed up. This attribute is turned on whenever you create or change a file.

Some people refer to attributes as flags because attributes are on/off switches. Pull the flag, the attribute is turned on. Pull the flag again, the attribute is turned off. Some attributes are automatically set by your programs and DOS. Others are set manually.

For example, the discussions on backup in Chapters 3, 15, and 16 describes programs that selectively back up only files that have changed. These programs examine the archive attribute and accordingly choose files with this attribute turned on. Another attribute of interest is read-only, which can be set manually through a program provided with DOS V3. When you have set this flag, the file is protected from erasure or alteration. This flag is useful in protecting nonchanging files. Attributes are discussed again later in this book.

The FORMAT command assigns attributes to several files. When FORMAT is commanded through the */S* switch to place the operating system on the disk, the command copies to the disk the three critical parts of DOS: IBMBIO.COM (or BIO.SYS), IBMDOS.COM (or MSDOS.SYS), and COMMAND.COM. The first two files are marked as both system and hidden files. These attributes prevent anyone from accidentally erasing or altering these essential start-up files.

With another option of FORMAT, the */V* switch, DOS gives your disk an electronic name, which you specify. The electronic label is maintained in the starting directory, but the label has no entry in the

FAT. Hence, the label reduces by one the number of files the starting directory can hold but takes up no additional disk space.

Understanding How DOS Names Disks

You should know the conventions used by DOS and in this book. Every disk drive has a name, which is a letter of the alphabet. When typed on the screen, the letter is followed by a colon. The first disk drive is drive A, entered as *A:*; the second is drive B, entered as *B:*; and so on, in ascending alphabetical order.

The usual convention is to have the start-up floppy disk drive (the floppy disk drive from which DOS can start) called drive A. This drive is typically the farthest right or the top disk drive. A second floppy disk drive is called drive B.

When the IBM PC XT was released, many users were surprised to learn that the hard disk drive was called drive C, even though the system had only one floppy disk drive. The convention IBM created works well, however. The first hard disk, regardless of the number of floppy disk drives, is drive C. A second hard disk drive is drive D. The A: and B: stand as floppy disk drive names.

If you use an external microfloppy disk drive, you'll note that this drive is typically drive E. Although most systems do not have this fifth disk drive, the suggested name is drive E.

Remember that the letter representing each disk drive is a convention, not an inviolate standard. You can have three floppy disk drives (A, B, and C) and use drive D to represent your hard disk. However, the assumption in this book is that A and B are the first and second floppy disk drives, with A being the start-up floppy disk drive. The first hard disk is drive C, and an additional hard disk is drive D.

Creating and Using File Names

Every time you create, delete, or edit a file, a document, a spreadsheet, or a graph, you refer to the file by its name. Any time you execute, run, or save a program, you use a file name. This section reviews the rules and conventions for naming a file in DOS.

The rules are simple. A file name in DOS has at least one and at most two parts. The mandatory first part is called the *root name*. This part is one to eight characters long. The optional second part,

the *extension*, is one to three characters long. When a file name has both a root and an extension, a period separates the two parts. For example, MUSIC.BAS is a BASIC program that comes with DOS. The root name is MUSIC, the extension is BAS (BASIC), and the two are separated by a period.

Certain characters cannot be used in file names. The following characters are permitted in a file name:

1. The letters A to Z (Lowercase letters are transformed automatically into uppercase.)
2. The numbers 0 to 9
3. The following special characters and punctuation symbols: $ # & ! ( ) { } ' - ~

The following characters are not permitted in a file name:

1. Any control character, including Escape (27d or 1Bh) and Delete (127d or 7Fh)
2. A space
3. The following special characters and punctuation symbols: ^ + = / [ ] " : ; ? * \ < > |

Each file has a unique name. If you change even one character of the name, DOS considers the new file name as referring to a different file. The contents of the two may be identical, but DOS "sees" the files as different. For example, MUSIC.BAS, MUSIC.BAT, MUSIC.COM, and MUSIC are all recognized by DOS as separate files.

Certain conventions for creating extensions allow some shortcuts for organizing and specifying files. A file name extension may indicate the purpose or contents of the file. Many programs always use certain specific extensions, and these common extensions are a handy method for categorizing files. The BAS for BASIC programs is an example. Microsoft BASIC, supplied with DOS, automatically adds the BAS extension when you save the file. When recalling the file through BASIC, you need not type *BAS*. BASIC automatically appends the extension and uses the result as a file name. In the same way, 1-2-3® automatically appends WK1 (for version 2.0) or WKS (for version 1A) to the root name you specify. When you retrieve the worksheet, the program looks only for files with that extension.

In general, for file names you create, you have the freedom to use any extension, or none at all. BASIC programs and 1-2-3 worksheets can have extensions different from BAS, WKS, or WK1, but you lose the convenience of not having to type the extension. More important,

the programs lose the built-in conventions inherent in the file extensions.

Some programs, however, limit certain files to certain extensions. Batch files, discussed in Chapters 7 and 8, *must* have the extension BAT. For a program to execute directly under DOS, the program must have an extension of either COM (a machine-language *command* file) or EXE (a machine-language *executable* file). You have no choice. The specific extension is required. In these instances, the optional free-form extension turns into an absolute rule.

Even in cases where you have no restrictions on file names, following conventions helps you organize and identify files. One common convention for word-processing text files is using DOC (documents) or TXT (text). Some people use LET for letters and MMO for memos. Whatever conventions you use, stick to them. Established conventions provide vital clues in locating a missing file. When copying, deleting, or moving files, you need to remember only the root name. You don't need to stop and think what extension should be given to a file; your fingers instinctively type the correct name. (See table 2.1 for common and mandatory file name extensions.)

Although less frequently used than extension conventions, the same or similar root names also can identify a file's "family," that is, show which files are related. For example, a batch file to invoke NewWord, a word-processing program, can have the same root name as the program. NW.BAT, the batch file, invokes NW.EXE, the program. Without additional information, an educated guess is that the file named NWINSTAL.EXE is the installation (INSTAL) program (EXE) for NewWord (NW). In the same way, you can easily guess that GO.DAT is the data file for the program GO.EXE.

The key to good file-name organization is to establish and use conventions for both the root names and the extensions. You can eliminate burdensome guesswork and head scratching by sticking to those conventions.

Reserved File Names

Certain names are called *reserved names* and must be avoided in your file names. DOS associates special meanings with these names because they are actually names of devices. One reserved name you'll frequently see in this book is CON, which means *console* (the keyboard and video display).

Table 2.1
Common and Mandatory File Name Extensions

Extension	*Stands for:*
.ASM	Assembler source file
.BAK	Backup file
.BAS	BASIC program file
.BAT	Batch file
.BIN	Binary program file
.C	C source file
.COM	Command (program) file
.DAT	Data file
.DOC	Document file
.DTA	Data file
.EXE	Executable program file
.HLP	Help file
.KEY	Keyboard macro file (ProKey)
.LET	Letter
.LST	Listing of program (in a file)
.LIB	Program library file
.MAC	Keyboard macro file (SuperKey)
.MAP	Linker map file
.MSG	Program message file
.OBJ	Intermediate object code (program) file
.OVL	Program overlay file
.OVR	Program overlay file
.PAS	Pascal source file
.PRN	Listing of a program (in a file)
.SYS	System configuration or device drive file
.TMP	Temporary file
.TXT	Text file
.$xx	Temporary or incorrectly stored file
.WKS	Lotus 1-2-3 worksheet file
.WK1	Lotus 1-2-3, Release 2, worksheet file

Trouble occurs if you use a reserved name as the only root name. When you use a reserved name as the root name, DOS "insists" that you mean the device, not a file. For example, if you save a file with the name PRN.TXT, the file goes shooting off to the printer and is

not saved on the disk. If you try to read a file named NUL, which is the name of the null device (no device), you get nothing. The extension does not matter. When you save a file to NUL."anything," you get just that—nothing. The penalty for violating this rule is loss of information, loss of system (a lockup), or both. Table 2.2 lists the reserved names and the device associated with each name.

Table 2.2
Device Names

PC DOS	
Name	*Device*
CON:	The video display and keyboard. As for generic MS-DOS, input from CON: comes from the keyboard; output to CON: goes to the video display.
AUX: or COM1:	The first asynchronous communications port. If a second asynchronous port is available, it is called COM2:.
LPT1: or PRN	The first line or parallel printer. Like LST:, this device is used only for output.
LPT2: LPT:3	The second and third parallel printers.
NUL:	The null device; used for testing.
Disk and Advanced BASIC	
Name	*Device*
KYBD:	The keyboard of the system (an input-only device).
SCRN:	The video display (an output-only device).

This caution does not mean you shouldn't use reserved names. Just use reserved names when appropriate. In Chapter 4, the setup chapter, you will see how to use a device name with the COPY command.

Wild Cards, the Secret of Powerful Commands

Two characters play a special role when you are specifying the file you wish to use. These characters, called *wild cards*, are the asterisk (*) and the question mark (?). By using wild cards, you can refer to a group of files with just one file name. Each wild-card character has its own meaning:

? matches any single character in a file name
* matches any number of characters in a file name

For instance, suppose that you want to copy from drive A to drive B all the files with a DOC file-name extension. The following single command captures all needed files:

COPY A:*.DOC B:

This command translates as "On the disk in drive A, find every file that has an extension of DOC and *any* (*) root name. Copy those files to the disk in drive B." You do not need to type a separate command to copy each file. One command covers all the files.

This first example is easy because it calls for files with a particular extension (DOC). By naming the documents with a DOC extension, you can copy all documents with a single command.

Seldom is life so easy. Suppose, instead, that you wish to copy only the worksheets for a specific financial plan, but you have other worksheets in the same directory. How can you copy only the needed worksheets when all the worksheets have the same WK1 extension?

Foresight in naming your files pays off. If you have given each worksheet a similar root name, such as FINPLN01, FINPLN02, FINPLNC, or FINPLNT, you can use wild-card characters to create a single file name to cover all the files you need, for example,

COPY A:FINPLN??.WK1 B:

This command copies all files with a root name consisting of *FINPLN* followed by any two remaining characters. The command calls files with names like FINPLN01 to FINPLN99, FINPLNAA, FINPLNTA, FINPLNC, or FINPLN. Notice that FINPLNC and FINPLN are included. The wild cards match the nonexistent characters in the shorter file names.

Most DOS commands allow wild cards in file names, but some DOS commands and many programs do not. The commands and

programs that do use wild cards in file names are pointed out throughout the book.

Remember that file names that follow conventions allow the most powerful use of wild cards. All that is needed is some ingenuity and consistency.

Understanding the Rule of Currents

You need to understand one more concept before you move on. You might call it the *rule of currents*. The rule is simple: If you don't specify a disk drive name to DOS, DOS uses the current disk drive.

Your computer has a current disk drive, the disk drive to which DOS *defaults* if no disk drive is specified. The A in the A> shows that drive A is the current disk drive. When the system prompt is C>, the current disk drive is C.

For example, to run the check disk program, CHKDSK.EXE, you type the command after the prompt and press Enter:

```
C>CHKDSK
```

DOS looks for the program CHKDSK on the current disk drive, C. After finding CHKDSK.EXE, DOS runs the program. Suppose that CHKDSK.EXE is on a diskette in drive A and not on the hard disk. To run the command, you type

```
A:CHKDSK
```

By placing the drive letter before the command name, you instruct DOS to use a different disk drive. If you do not include a disk drive name, DOS searches the current disk drive, C, for the program. DOS then correctly informs you

```
Bad command or file not found
```

This message tells you that you either mistyped the name (not true) or the disk-based file holding the command program cannot be found (true).

If you omit the drive letter, DOS relies on the default value, which is the current disk drive. DOS has more than one "current" item. This subject is covered in depth in Chapter 5.

Summary

This chapter has presented some background on DOS, told how DOS is divided, discussed how DOS divides a disk, and explained what housekeeping areas DOS places on a diskette. You have also learned about file names, wild cards, and defaults. The key points to remember are the following:

- A disk operating system is the fundamental set of programs that gives you and your applications programs control of the computer's resources.
- PC DOS is the IBM implementation of MS-DOS. The two are closely related and compatible but not identical.
- DOS consists of four parts: the BIOS, BDOS, command processor, and housekeeping utilities. The first three are the core of DOS and reside in your computer from the time you start up until you turn off the machine
- All disks must be prepared before use, undergoing physical and logical formatting. DOS performs both types for a floppy diskette but only logically formats a hard disk. An outside program must be used for physically formatting a hard disk.
- DOS divides a disk into smaller units for faster, easier handling. Hard disks are divided into cylinders, or tracks, and sectors, with the sector being the smallest unit of storage. A cluster is the smallest unit of disk storage DOS can allocate and can include one or more sectors.
- DOS establishes several housekeeping areas on the disk. During operation the directory and file allocation table (FAT) are the important areas.
- Disk drive names are letters. The letters start with A and ascend by one for each disk drive. A and B are used for floppy disk drives; C and D are hard disk drives. When you enter the drive names on the screen, they must be followed by a colon (:).
- File names consist of two parts: a one- to eight-character root name and an optional one- to three-character extension. The root name and the extension are joined by a period.
- A disk drive name can precede a file name. No space appears between the drive name and file name.
- Devices have reserved names. You may not use a device name as the root name of a file.

- The wild card characters * (match all) and ? (match one character) are allowed with many DOS commands. With these characters, a single file name can match many files.
- DOS has a current disk drive. DOS searches this disk drive for a file when no disk drive name is specified.

The next chapter presents information you need in order to buy a hard disk. That chapter defines terms and explores the meaningful and meaningless characteristics you'll encounter when comparing hard disks. If you have already chosen your hard disk, skip to Chapter 4 on setting up your hard disk.

Chapter 3

Selecting Hard Disks

Today hard disks have become very much a commodity item. The commodity approach has both pluses and minuses. The big plus is the hard disk's recent drop in price. Another plus is that disk drives are available from a wide variety of sources. A big minus for the state of hard disks is that lower quality drives are entering the market. The other disadvantage is confusion. You can scarcely turn an ad page in a computer magazine without seeing a hard disk drive for sale.

How do you decide which hard disk to buy? For most people, the choice is simple: a fixed (Winchester) disk drive. One major reason is that this hard disk drive is included with many computers. But even those who are fitting a hard disk to an older PC tend to choose the fixed Winchester hard disk.

You might ask, "If you have already given the answer, why discuss the question?" The fixed disk is the best choice for over 90 percent of all personal computer owners. However, even if you choose a Winchester, you still have a number of issues to consider. This chapter discusses those issues.

If you already own a disk drive, skim this chapter and move to the next, which deals with setting up your hard disk. If you're exploring the market for a hard disk, are just interested in the choices you will have later, or are a "vanilla PC" (not XT or AT) owner, this chapter is for you.

The issues to examine when you are purchasing a disk drive are the following:

- Should I get an internal or an external disk drive?
- Is this a first hard disk, an additional hard disk, or a replacement?
- Which disk specifications are meaningful and which are meaningless?
- How do I balance performance and price?
- What additional hard disk drive features should I consider?
- What type of backup will I use?

This chapter introduces each issue, examines terms and meanings, and gives guidelines for selecting a hard disk. Because several issues are intertwined, you should read all of this chapter before making your final decision.

Choosing IBM or Non-IBM

A single theme runs through any disk purchase: "Should I buy the hard disk from the computer manufacturer or another source?" The answer is a definite "maybe."

When the first IBM-sanctioned hard disk machine came out (the XT), most people would contemplate buying nothing but a hard disk drive with the IBM label. This choice still is usually the "safe" but more expensive way to choose a hard disk.

Even the safe choice, however, is not always best, as was proved when owners of the first round of Personal Computer ATs reported disk drive problems. Until 1985, IBM did not manufacture disk drives for its Personal Computer family. IBM bought disk drives from respected firms like Seagate Technology and MiniScribe Corporation.

When the AT was introduced in 1984, IBM used drives from a small California company called Computer Memories Incorporated (CMI). Problems surfaced and customers complained that data was being corrupted on the disk drive. The exact cause is not known, but most industry watchers believe that it was a combination of problems with PC DOS, the fixed disk adapter board (from Western Digital), and the CMI disk drive. In late 1985, IBM did not renew its contract with CMI and started using its own disk drives.

The story points out that even IBM has not always used IBM disk drives. Using non-IBM disk drives in a personal computer is "socially

acceptable." Other disk drives can be less expensive and as reliable as IBM disk drives. The key is knowing what to look for in a hard disk drive.

Many people choose IBM because of the company's reputation for good service and repairs. Several other computer manufacturers have excellent networks for customer support. However, if your hard disk needs service, you may very likely have your dealer (not the drive manufacturer) do the repairs.

Don't limit yourself to just IBM equipment; too many doors close quickly. On the other hand, don't exclude IBM, or the other computer manufacturers, from your disk drive purchase. Aspects of IBM, or the computer manufacturers, and local dealer support offer value for the money.

Putting the Disk Inside or Out

When you get ready to buy a hard disk, the first major decision is where you'll put the hard disk. An *internal* disk drive fits in the computer's system unit. An *external* disk drive is a free-standing unit.

If you are buying a microcomputer and a hard disk at the same time, this decision is almost unnecessary. Most computers have provisions for one hard disk in the computer unit; and some can handle two hard disks. The internal hard drive usually makes the most sense. The decision of internal versus external is a more open question when you are adding the first hard disk in a PC or the second hard disk in an XT- or AT-like computer.

The basic issues are physical room, available adapter slots, and sufficient power. All three must be balanced against cost.

Room for the Drive

Do you have room enough in your computer for a hard disk drive? This answer will determine not only whether you should buy an internal or external disk drive but also whether you should consider an arrangement of half-height disk drives.

If you have a PC or an XT with one full-height floppy disk drive (or two half-height floppy disk drives), you can easily mount an internal hard disk drive in the slot where a full-height disk drive goes.

An AT has enough room for one full-height and three half-height disk drives. Adding the first hard disk drive is easy. That unit fits in

the bay behind the cover of the machine. The second hard disk fits under the floppy disk drive at the right of the computer. Adding a second disk, providing it's a half-height disk drive, allows for two half-height floppy disk drives. A second full-height hard disk drive leaves space for only a single half-height floppy disk drive.

The choices are more difficult for a PC with a floppy disk drive. Placing an internal hard disk in a PC with two floppy disks ordinarily means replacing at least one floppy disk drive.

You can (1) remove one floppy disk drive to make room for the hard disk; (2) replace both full-height floppy disk drives with half-height models and use a full-height hard disk; or (3) replace one floppy disk with a half-height floppy disk and use a half-height hard disk.

Using an external hard disk eliminates the need to replace any floppy disks. External hard disks are usually easier to install than internal hard disks, but installation is really a minor consideration. The downside to external disk drives is that they require more desk space, something that may be a premium in your particular setup. An external hard disk also costs more (about $150) because the disk drive has its own housing and power supply.

If your computer has no more bays for an internal disk drive, the recently developed *hard-disk-on-a-card* or *hard-card* may help you. This device has a Winchester disk with its electronics mounted together on an adapter board. The card fits into a vacant expansion slot. The advantage to the hard-card is that the card can be installed without interfering with your floppy disk drives. Installation is convenient. Just open the computer cabinet, pop in the card, and you are just about off and running. Many people have bought these hard-cards for portable personal computers (like Compaq®) because of the ease of upgrading.

However, the hard-card is not for everyone, as is explained in the sections on adapters and power.

The Adapter

All disk drives require an adapter. Some people call this adapter an interface card. Technically, most adapter cards inserted in the computer for the Winchester disk drives are the hard disk *controller card.*

If you are purchasing the first hard disk for any computer but an AT, you'll need to purchase an adapter with your disk drive. Because the AT uses a combination floppy/hard disk adapter, a separate

adapter is not necessary. Be sure to include the cost of this adapter when figuring the cost of your hard disk. The average adapter price is between $150 and $250.

All adapter cards are not created equal. As IBM demonstrated, some combinations of controllers and Winchester hard disks yield unsatisfactory performance. Some controller cards are more compatible with IBM's adapter than others. If you are buying the controller card in the "open market," get concrete assurance that the controller works with your selected disk drive and is compatible with IBM's adapters.

The adapter requires one expansion slot inside the computer. If you have a slot free, you can install the adapter card. Some adapter cards, however, are thicker than the 1/2-inch wide space given to each expansion slot in the XT and AT.

Some hard-cards extend over two full expansion slots. Other hard-cards allow a half card (a shorter expansion card) to be used next to the hard-card. If you have two expansion slots free, any hard-card can be used. If you have only one expansion slot free, be careful in your selection of a hard-card. You'll need the type that takes up only one slot, or you must rearrange your cards so that a half-size adapter card can be used next to the hard-card.

Power

An internal hard disk, including the hard-card, draws its power from the computer's power supply. The various hard disks have a wide range of power requirements. Some disks use less than 10 watts; a few units require over 100 watts. The computer's power supply must have sufficient current for the motherboard, all expansion cards (including the hard disk controller card), and the hard disk.

Power is not an issue on the AT and most XTs. The power supplies in these machines (210 and 135 watts, respectively) are sufficient to drive most combinations of expansion cards and hard disk. The rare XT owner whose expansion cards are power-hungry may have problems. The culprit will be an unusual expansion board, such as a video-image capture board, or a several-unit set of expansion boards that control a specialty device.

Power requirements are an issue with the PC, whose power supply is rated at 63.5 watts. Several combinations of expansion cards and hard disk drives can quickly overload the supply. The frequent

culprits are multifunction memory boards or expanded memory boards (EMS).

Erratic operations are symptoms of insufficient power. You may experience sporadic read or write errors on any disk drive, RAM memory errors (usually parity errors), and occasional lockups of the computer (with or without an error message). These problems occur with no apparent pattern or reason. Problems occur more frequently when the computer is warm than when it is cool. Warm circuits offer more electrical resistance and consume slightly more power. Marginal circuits may function normally when cool but malfunction when warm. The heat generated by the internal hard disk also aggravates the problem.

The power supplies of the PC family have an overload protector. This arrangement prevents the power supply from burning out and critical circuitry from sustaining damage. However, computer operations halt when the protector kicks in.

If the power supply is too low, your options are replacing the supply or using an external disk drive. Replacement power supplies run between $75 and $150, excluding installation. With the cost of an external disk drive running about $150 to $200 above the cost of an internal disk drive, the choice is simply a matter of preference. If your PC is under warranty, be aware that installing a non-IBM replacement power supply voids the warranty. Because most PCs are more than one year old, however, this question is not usually an issue.

Before pitching your power supply, check the fan and vents on the computer. Clogged vents can slow the flow of cooling air, causing marginal chips to malfunction. Clean the vents and fan and make sure hot exhaust is not being trapped. Pushing the back of the PC against a wall or desk, for example, aggravates thermal problems.

For the record, most hard-cards consume less power than internal fixed disks. Where a normal internal Winchester causes an overload, a hard-card may work.

First or Second Drive

Are you installing the first hard disk for the system, a second disk drive to supplement the first, or a replacement for a disk drive already in the computer?

The issues of space and power have already been covered for the first hard disk for your system. Simply remember that you'll need an adapter card for the computer (unless you have an AT or are

purchasing a hard-card). The AT uses a single adapter, which handles the floppy and hard disk; no additional board is needed. The card in the hard-card *is* the adapter card. Be sure to add the price of the adapter to the total cost of the hard disk.

If you're adding a second hard disk to a computer other than an AT, you're probably looking for an external hard disk. Unless you can purchase (or have purchased) half-height hard disk drives, you have no room for the second hard disk. Personal Computer AT owners can use two full- or half-height hard disks.

Hard-cards are designed to be the primary (first) hard disk drive of the computer. Some of these devices cannot be used with other hard disks. If you plan to add another hard disk later (which few people do), make sure that your hard-card can adapt to sharing responsibilities with another hard disk. Some hard-cards also are unsuitable as secondary hard disk drives.

If you're replacing a fixed disk, you might consider stepping up to a higher capacity fixed disk. Full- and half-height fixed disks come in the 20- to 80-megabyte range and higher, and take no more space than the 10-megabyte hard disks. These higher capacity fixed disks place little additional strain on the power supply and can easily replace lower capacity disk drives. However, the Personal Computer family and DOS place some restrictions on the size of the disk drive. This problem and its solution are discussed in the section "Capacity."

The Choice

The choice between an internal and external hard disk should not cause great anxiety. If the computer physically and electronically can accommodate a hard disk, the choice narrows down to price and convenience. For most buyers, the choice of an internal hard disk is easy. Because the XT and AT have bays for the disks, the internal disk drive is best. The only reason to reverse this decision is that a particular hard disk you pick cannot fit inside your computer.

Unusual circumstances may force some XT owners to purchase an external hard disk. The need to preserve previous investments in floppy disk drives or the power requirements of unusual expansion boards influence these owners to buy the external hard disk.

An internal disk drive may or may not work in a PC. The principal reason for failure is an insufficient power supply due to the use of power-hungry expansion cards. You need to total the wattage required by each expansion board, the system board of the computer, the floppy disk drive(s), and the prospective host disk. Each

expansion board manufacturer publishes this information in the board's documentation. If the information is not readily available, the best alternative is a "test drive," installing the selected hard disk and watching the results.

Where a normal hard disk will not fit, a hard-card may. It consumes less power, but some hard-cards overlap an additional expansion slot (in some cases, half of an adjoining slot). If expansion slots are at a premium, you should choose the straight and narrow (a thinner hard-card that does not cover additional expansion slots).

Under one circumstance, an external hard disk is the choice: the preferred hard disk unit cannot fit into the computer's system unit. When considering external units, you simply balance features against price and desktop space needed for the external unit.

Many drives using removable media do not fit inside the system unit. An example is the Bernoulli Box (see figure 3.1), whose drive mechanism is too large to fit in the system unit. The Bernoulli Box uses a removable hard cartridge that houses a flexible media. The media becomes "frigid" (rigid) when in use and can hold up to 20 megabytes. The speed and storage features of the box are comparable with many hard disks on the market. Because the Bernoulli Box is often discussed in terms of its backup features, a more complete description is presented in Chapter 16.

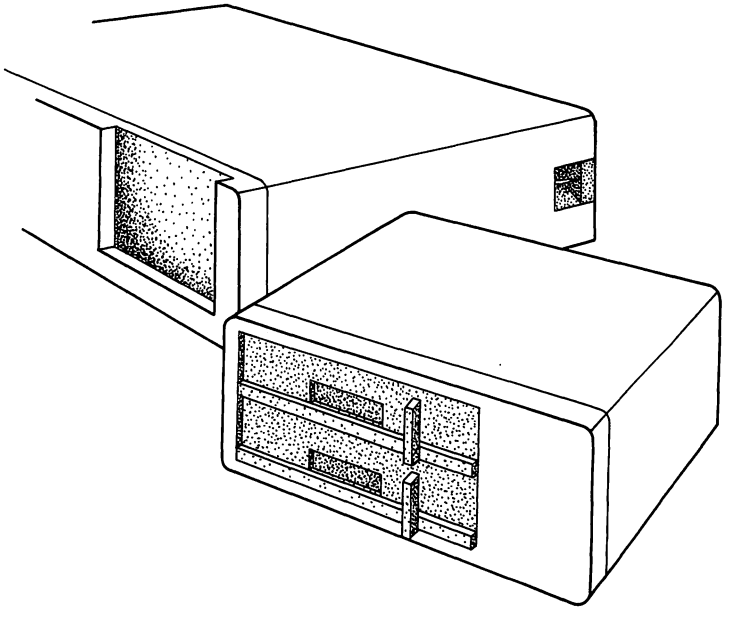


Fig. 3.1. *The Bernoulli Box.*

Other manufacturers also use removable media, and some produce both internal and external units. These manufacturers include SyQuest Technology (a true removable Winchester hard disk), IDEAssociates (also removable Winchester drives), and a joint Kodak/ Data Technology venture (a removable floppy-type medium). Some manufacturers bundle the external hard disk unit with a removable cartridge tape drive.

Making Sense of the Stats

Manufacturers love statistics; but as a buyer, view these statistics cautiously. The problem with statistics is not what they are, but how the statistics are used. A friend once quipped that through statistics he could prove that Rhode Island was bigger than Texas. He almost won the argument by using some convoluted logic.

The purpose of this section is not so much to give you an in-depth technical education in hard disks as to provide you with enough information to understand the meaningful differences among hard disks. The differences extend to more than megabyte capacity. This section discusses several key statistics and the role of each in the selection of the hard disk. The most important statistics concern capacities and speeds.

Capacity

Of course, hard disk capacity is nothing to be sneezed at. When the IBM XT first came out, 10 megabytes of hard disk storage seemed like a great deal. Then the IBM AT was launched with 20 megabytes as standard hard disk capacity, and early in 1986 IBM boosted standard capacity to 30 megabytes. Disk manufacturers and users began to talk about "mass storage." The best definition of mass storage seems to be "never massive enough." People easily fill whatever storage capacity they have. As hard disks with several hundred megabyte capacities are introduced, we stop saying we'll never use that much capacity. We just note when we think others will scream for more.

Both your computer and DOS restrict the size of the disk drive you can use. The restrictions are surmountable by third-party (non-IBM) efforts, however.

The first restriction is computer recognition: Does your computer recognize a certain hard disk? The problem lies in the ROM BIOS,

the software in read-only memory on the computer's motherboard. The ROM BIOS of the PC recognizes only hard disks with capacities of 10 megabytes. The ROM BIOS of the XT recognizes only 10- or 20-megabyte disks. The ROM BIOS of the AT seems to know 20- and 30-megabyte hard disks and selected others but has amnesia with other hard disks. The latter case requires third-party software to remedy the situation.

Another deficiency lies in the disk operating system, DOS, which does not recognize disks with more than 32 megabytes of capacity. This limit forces the user into one of two situations. Through software provided with such storage giants, the physical hard disk is divided into two or more logical (apparent) units, each with a capacity under 32 megabytes.

Alternatively, some manufacturers provide (or outside publishers offer) special software that modifies DOS to handle the larger capacities. Both solutions work well; however, forward-thinking people hope Microsoft and IBM will modify later versions of DOS to handle these larger capacity hard disks.

Speed

Another important statistic is speed. Speed is not a single statistic but several. Speed can be broken down into access time, seek time, settle time, and latency.

Of these statistics, access time is bandied about most often. *Access time* is the elapsed time from when the operating system requests a read or write operation to when the data is ready for transfer. The time unit is milliseconds (ms), and access times range between 18 ms and 100 ms.

The total access time can be broken down into three components: *seek time*, the time required for the disk heads to move to the right track; *settle time*, the time needed for the head to "settle down" on the given track; and *latency*, the time required for the right sector to swing around to the recording heads. Latency is rarely factored in by disk drive manufacturers when giving their statistics.

The most important statistic overall is *average access time*, the statistically average time for the disk drive heads to move from one random point on the disk to another random point. This time is the common statistic for rating overall disk speed. The other statistics mentioned are minor.

You can infer from the preceding explanation that faster access times are better. For the most part, that inference is correct.

However, you usually have to pay more for a disk drive with a significantly faster access time. The price difference is related to the mechanism that controls the arm holding the recording heads.

Drives at the low end of the price scale rely on *stepper motors*, which are controlled by an older, less expensive, heavily mechanical technology. Stepper motors also are used in floppy disk drives. The shaft of the stepper motor does not rotate in a smooth continuous motion but by small even increments, hence the name *stepper*. In a hard disk, the access time of stepper motors is about 60 ms to 100 ms.

The more expensive, faster drives use a more sophisticated technology based on the same principle as audio speakers. *Voice-coil* technology is more electronic than mechanical. The coil is housed in a stationary magnet. The recording head arms are connected to the coil. Regulated electric current changes the magnetism in the coil, moving the coil and recording arms back and forth. Voice-coils are both faster and more reliable than stepper motors. The access time for voice-coils is usually from 40 ms down into the teens. Because the voice-coil has fewer moving mechanical parts, it is more hardy.

Bear in mind that if you are comparing access times of different capacity disk drives, the statistics can "lie." Given a 60 ms access time for both a 10- and a 40-megabyte disk drive, the 40-megabyte drive is faster. By definition, access time is the rate of the random point-to-point jump. The 40-megabyte drive is randomly jumping in a larger area than the 10-megabyte drive. To achieve the same access time, the 40-megabyte disk drive must be faster. The apparent difference between the two is minimal and serves as a demonstration of the imaginative "specsmanship" in which manufacturers indulge.

Is there a best access time? Is a 28 ms hard drive worth the difference in price from a 40 ms, 60 ms, or 85 ms disk drive? The difference between 28 and 40, 40 and 60, or 40 and 85 milliseconds does not sound significant. True, the difference in speed is not major for everyone, but the difference in price can be several hundred dollars.

The best guidelines are based on use: heavily disk-bound operations can benefit by fast access times. For example, database applications are heavy "disk hitters." A record is read from the disk, modified, and written back out to the disk; then another record is read, modified, written to disk; and so on. When the system is constantly accessing the disk for information, you may need faster access times.

For two reasons, AT owners should stick to disk drives at 65 ms and under. First, IBM's fixed disk drive is rated at 40 ms. Some AT-clones use disk drives as fast as 18 ms. Almost one-third of the 500 percent improvement an AT has over an XT or PC comes from the faster hard disk. As the AT often serves as the workhorse in a network or other disk-bound application, don't limit your AT's throughput by using too slow a disk drive. Second, the next major release of DOS, DOS 5, is a multi-tasking operating system, meaning that you can run more than one program at a time. When running a multi-tasking operating system, the major drag on computer throughput is the disk drive. Using a slow drive will significantly slow the performance of DOS 5 and your programs.

At the other extreme are people who infrequently use a hard disk during the work day. If you spend 90 percent of your time modifying a spreadsheet or processing a document that is contained in the computer's RAM memory, not hard disk space, you may not need the faster access times. In your final decision, the weight you give to access time should be determined by your personal computing needs.

Transfer Rate

Another important (but not important) factor contributing to a hard disk's speed is the *transfer rate*. Here is the contradiction.

The transfer rate represents the number of bits of data transferred per second between the hard disk and your computer after the drive head has reached the place where the desired data is located. Because the faster the transfer rate, the faster the overall disk performance, transfer rate is important.

The reason that transfer rate is not as important as it seems is that most disk drives use a standard type of interface, which has a top rate of 5 megabits (or 625K bytes) per second. Almost all hard drives use this standard, which sets an upper limit on transfer rates of the hard disk. Hence, transfer rate is an interesting subject for reading, but a meaningless statistic for comparison of these drives.

Recent advents, however, will contradict the statement that all disk drive transfer rates are equal. Explaining the why's of this contradiction requires some additional information.

The original interface for most hard disk drives was patterned after the IBM approach to I/O channels for its 8-inch Winchester disk drives by Allen Shugart of Shugart Associates. The interface was called the Shugart Associates System Interface, SASI (pronounced "sassy"). Allen Shugart later left Shugart and formed Seagate

Technologies; so the name SASI is often attributed not to Shugart Associates but to the newer company, Seagate.

Operating under the jurisdiction of the American National Standards Institute (ANSI), a disk drive industry group formed the X3T9.2 Committee. The committee improved and "standardized" the SASI interface, but ANSI rules prohibit keeping the Shugart name. So the committee approved the Small Computer System Interface, SCSI (humorously nicknamed "scusy").

Almost all hard disk drives for today's microcomputers use the SCSI interface. The SCSI interface applies not just to disk drives but to all intelligent peripherals, such as printers, modems, and local-area networks. The interface specifies physical and electronic connections and software protocols (commands) used to control the devices.

The Enhanced System Device Interface (ESDI, nicknamed ed-see) is a higher speed electronic interface than SCSI. Current ESDI specifications use a 10-megabit transfer rate, twice as fast as SCSI. A recent ESDI specification boasts a 15-megabit transfer rate, three times faster than current hard disks. ESDI's role in microcomputer systems is growing because of the Intel 386.

Big brother to the PC's 8088 and AT's 286, the 386 microprocessor runs two to five times faster than the 286. Because the processor handles data at a high rate of speed, the 5-megabit transfer rate of an SCSI disk drive is too slow. Such disk drives waste the capabilities of a 386 system. Rather than wasting the 386's potential, Compaq's first 386 computer, the Deskpro 386, uses the ESDI interface and appropriate disk drive. The 10-megabit transfer is more suitable to the higher performance machine.

Should you purchase an ESDI interface and disk drive? Unless you use a 286 or 386 system, the faster ESDI drive is an over-expensive solution. The simple choice of a faster access-time SCSI disk drive is more appropriate. Today, ESDI interfaces and disk drives are almost twice the price of the average SCSI disk drive. 8088-based systems do not have the horsepower to exploit the faster disk drive.

Most PC AT's would receive only marginal benefits from ESDI drives. The preferred choice for a 286 running at system clock speeds under 10 megahertz (MHz) is the faster SCSI drives. The 286 systems that run over 10 MHz and 386 systems (which start at 12 MHz) can benefit from ESDI drives.

At this writing, operating systems do not recognize the ESDI interface without appropriate device driver software. Unless your system recognizes an ESDI drive, you will not be able to boot DOS from the drive.

ESDI provides half of the solution (the hardware half). Another standard, the Intelligent Peripheral Interface or IPI, covers both hardware connections and software protocol (commands). IPI controls intelligent peripherals, such as printers, modems, and disk drives. When the term IPI is bandied about in hard disk ads, you'll know the hard disk conforms to this hardware and software specification.

With price and convenience factors in mind, ESDI drives are not yet the preferred choice for PCs and ATs. As we enter the 386 generation, there will be a pronounced difference between the "standard" disk drives of today (SCSI) and the "standard" of tomorrow (ESDI/IPI). Today, all disk drives "look alike" and transfer rate is unimportant. By 1988 when ESDI disk drive prices reach today's SCSI prices, the transfer rate difference between SCSI and ESDI will be an important consideration.

Interleave

A quantitative factor in determining a hard disk's speed is the interleave factor. Although interleave is not a meaningful statistic for comparison, interleave is one of the lesser understood terms associated with hard disk and deserves an explanation.

Most hard disks do not write a file on consecutive pieces of the disk surface. Instead, the disk records some data on one sector on the track, skips around, records some more on another sector on the same track, skips around, and so on. A file on your disk is probably split into many different physical sectors. The *interleave factor* is the ratio of physical disk sectors skipped for every sector actually written on. An interleave factor of six means that one of every six sectors is actually written on.

Why would the hard disk manufacturers deliberately choose an interleave factor other than one, forcing more time to transfer data between the computer and the hard disk? The operating system of the computer needs some time to process the data before the data is written or read. To allow time for this processing, manufacturers deliberately choose an interleave factor that slows the data transfer.

Some hard disk and third-party vendors provide software that changes the interleave factor; for example, the absolute formatting routines for the fixed disk in the IBM Advanced Diagnostics. *Exceptional care should be taken if you choose to alter the interleave factor.* The best advice is "If it ain't broke, don't fix it" unless you have very good reason to do so.

The hard disk manufacturer has formatted the disk with an appropriate interleave. Only very experienced computer users who know that the factor should be changed should alter the interleave.

For instance, you might change the interleave because the computer's CPU speed has increased, you need to speed up the computer, you purchase a different disk controller, or you make a major change in the operating system. Naturally, the hard disk must be backed up before changing the interleave because the absolute formatting process destroys all information on the hard disk.

In any case, the interleave factor is not a factor in the purchase decision. Interleave is simply a parameter that has been fine-tuned so you can get the most out of the disk you use.

The Shock Treatment

Chapter 1 mentioned that strong physical shock is a serious danger to any hard disk. Many hard disk manufacturers are now testing their drives under "shock therapy" to determine how many Gs the drives can withstand without data loss. Although no minimum standard exists, the ratings give an indication of how well the drives are shock-mounted.

The common operational and at-rest ratings are 10 and 20 Gs, respectively. If all other factors are equal, choose a disk drive with a higher G rating. This advice is more important for portable computers than desktop models. Portables are jostled more frequently. The at-rest rating is important. Because portable computers are almost never used when moving, the in-use rating is less important. However, all factors in hard disk selection do not have equal weight. Understandably, the shock rating usually takes a back seat to price and performance.

Balancing Performance and Price

Only two performance statistics are truly important: megabyte capacity and access time. All other factors, transfer rate, G rating, and so on are marginally uniform between the disk drives. Capacity and access time are strongly reflected in the price of the hard disk.

Hard disks have no "right" size. You determine the correct size of the hard disk according to your need and the price. The bare minimum today is 10 megabytes. The largest unit available holds

about 384 megabytes. Chances are strong that your needs and budget dictate a size between the two extremes.

For a personal computer that only one person uses infrequently during the day, the 10- or 20-megabyte disk drive makes sense. This size is large enough to hold several applications (such as a spreadsheet, a word processor, or a small accounting program). Usually, the data files or documents used by the programs will take more space than the program itself.

If the computer is to be networked (the computer's disk drive is shared by more than one computer), 20 megabytes are minimum and more are preferred. Because more people use the disk drive, more storage is needed. If you use the computer for a large accounting application, 20 to 30 megabytes usually suffice. If you use several major applications or develop programs, 20 megabytes or more are desirable.

Fortunately, the incremental cost of increasing capacity is low. A 20-megabyte hard disk is not twice as expensive as a 10-megabyte disk, nor is a 30-megabyte disk three times as expensive as a 10-megabyte disk. For example, the following list of prices was pulled from some mail-order ads in *PC Week* for August, 1986. All prices include a hard disk drive from some major non-IBM manufacturer and a controller card (adapter).

Drive Size	*Price*	*$ per megabyte*
20 megabytes	$ 589	$29.45
30 megabytes	$ 659	$21.97
40 megabytes	$ 769	$19.23
72 megabytes	$1,295	$17.99

Note the incremental cost of $70 for 10 more megabytes (a 30-megabyte drive versus a 20-megabyte drive). You can get 20 more megabytes for just $180 (40 versus 20 megabytes). Although the most cost-effective disk drive is the 72-megabyte model, $1,300 for a hard disk drive is a steep price to pay for a capacity you may not really need.

The other important statistic is access time. The preceding list focused on full-height disk drives with 40 ms access time (the 72 megabyte drive has a 28 ms access time). By contrast, the following are the prices of half-height disk drives with 65 ms access times:

Drive Size	*Price*	*$ per megabyte*
20 megabytes	$429	$21.45
30 megabytes	$549	$18.30

The half-height drives are less expensive for two reasons. Half-height disk drives cost less to manufacture than full-height disk drives. Almost as important is the fact that these drives have slower access times than the previously listed drives. On a PC or XT, the difference in performance between 65 ms disk drives and 40 ms disk drives is barely noticeable. On an AT, 286 XT, or similar computer, the performance difference is noticeable but tolerable.

The choice is simple: choose the best disk drive you can afford. If you have unlimited money, bigger and faster is better. Most users' personal preference is to spend a little more money on storage capacity than on access time. The best news is that drive prices are spiraling down. By the time you read this book, the price you pay for a disk drive will be lower than those listed.

Considering Other Features of Hard Disks

You should consider a number of other features when selecting a hard disk drive. Most features can be viewed as conveniences, but some, such as automatic parking, may be more important.

This section discusses the major features: bootability, automatic parking, and the type of media used in the disk drive. Some frequently overlooked items for installing a hard disk are also mentioned. Finally, the issue of service is introduced.

Can You Boot It?

The first nonstatistical characteristic to consider is the capability to boot your computer system from the hard disk. When you turn on the computer or do a Control-Alt-Del (system reset), can you start DOS from the hard disk? This capability eliminates the need to start DOS from a floppy diskette and generally speeds the start-up process.

This feature sounds like an obvious one. In 1983, however, most add-on hard disks did not have this feature. Today bootable hard disks are standard. Any IBM XT or AT and all major work-alikes equipped with a hard disk automatically boot from the hard disk.

Only when you buy an add-on or nonstandard product should you double-check that the drive has this feature. For example, the early Bernouilli drives (see Chapter 16) were not bootable and required a floppy diskette with the appropriate DOS files in drive A to start the

computer. Although most people turn on or reset the system only a couple of times a day, having to use the start-up floppy reduces the fun and convenience of a hard disk. Most disk drive manufacturers, including Bernoulli, are on track today. The equipment to make the Bernoulli Box bootable is available as an option.

Can You Park It?

A desirable and occasionally essential feature is automatic "park and lock," usually shortened to the term *automatic parking*. To explain this feature, a short discussion of how the disk drive platter surface is used is necessary.

The entire surface of a platter is not used for recording information (see fig. 3.2). On the standard 5 1/4-inch hard disk drive, only a one-inch band near the center of the disk is used. This band is called the *data area*.

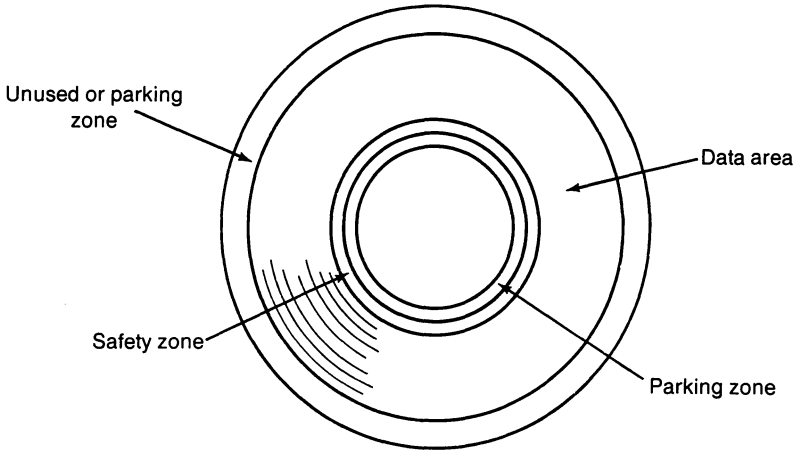


Fig. 3.2. *The safety areas of the hard disk platter's surface.*

Parking the disk drive heads means that the arm is pulled back so that the recording heads are not over the platter's data area. The heads are mechanically *locked* in this *safety zone*. If the heads should scrape the platter while parked, no information would be lost.

If your hard disk is equipped with automatic parking, whenever the system is turned off, the read/write heads are moved away from data areas and locked in the safety zone. This feature avoids accidental destruction of any data should the machine be jostled or bumped.

However, this feature usually is not essential because virtually every hard disk comes with software that parks the disk in the vacant spot. You must run the software, but the process takes only a few seconds.

Parking software for the hard disks initially installed in IBM equipment is on the Diagnostics diskette in the back of the Guide to Operations manual. The option from the main menu is called "Prepare system for relocation." Running this option parks the hard disk heads. When you turn on the computer again, the system moves the heads back into position for work.

For the portable computer, however, automatic parking can be an essential feature. Most portables use automatic-parking fixed disks. An automatic-parking disk is helpful for portable computers for two reasons. Forgetting to run SHIPDISK or a similar program before turning off the computer is very easy. Portable computers also get more physical abuse than desktop models. For these reasons, automatic-parking disks are a convenience and potential lifesaver to portable computer users.

What Is the Medium?

As mentioned in Chapter 1, the platters of the hard disk are coated with a metal oxide. The type of coating affects the robustness of the disk.

Most lower-priced models use an oxide-coated surface. The high performance models use plated platters. The stronger surface is more resistant to damage when a wayward drive head divebombs the platter (during shutdown or severe physical shock). The divebombing is known as a head crash, and the last chapter painted the gory details of such actions. As mentioned, thin-film platters are more resistant to head crash.

The fact that the disk drives use hardened media does not have a direct bearing on your decision because you cannot change the material. This factor is reflected in the shock rating of the disk drive and is rarely a top consideration.

What Little Surprises?

Murphy's law states, "If anything can go wrong, it will." When selecting your hard disk drive, don't forget the small, vital parts that can stall a hard disk installation. True to Murphy, most installation

problems seem to occur on a Saturday afternoon, turning the thrill of discovery into a weekend of frustration.

When purchasing a hard disk, make sure that you're getting all the cables and power cords you need. Most internal disks get their power from the power supply of the system unit. The PC has only two power connectors, one for each floppy disk drive. The XT and AT have three. If all power connectors are in use, you'll need an adapter that splits one connection into two. When you purchase an external disk drive, make sure that all AC power cables and connecting cables are included. Few things are as frustrating as trying to find these funny three-pronged AC power cables on a Saturday morning.

The AT's data cable (the wider of the two cables for the hard disk) has connectors for two disk drives. However, each disk drive needs a separate controller cable. A lack of this often-forgotten flat ribbon cable has delayed more hard disk installations than any other problem.

Another stumper may be the guide rails needed for installing the hard disk in an AT. These rails are screwed on the two lateral walls of the disk drive. The rails help hold the hard disk in the computer's drive bay. Forgetting the rails means the disk drive cannot be mounted correctly. You can do without the guide rails for a short time only—one or two days at the most. One good bump to the computer could terminate an otherwise healthy hard disk.

Incidentally, mounting a disk drive in an XT does not require an extra data cable (the computer has an idle cable) or the guide rails. The disk drive just bolts to the walls of the bay.

Aesthetic considerations may call for an extra faceplate to cover the hole left when you mount a half-height disk drive in a full-height bay. If there is a chance of a paper clip or similar metal object getting into the bay and shorting out the computer, make the new faceplate a mandatory item.

One last detail concerns the screws needed to mount the disk drive. Unless someone has removed these screws, most computers, including all IBM models, have the necessary screws in the right holes inside the system unit. If you are unsure, ask the vendor to add a couple of screws to your order.

The total cost of the data cable, guide rails, power cable adapter, extra power cable, and extra screws is seldom more than $35.00. When placing the order for the hard disk, ask whether any of these pieces are needed. It is better to sound like a fool when placing the order than to look and feel the fool when installing the hard disk.

Can It Be Serviced?

As has already been stressed, all hard disks fail. No one can say precisely when the failure will happen, only that failure will occur. Two statistics concerning disk failure enter into the manufacturer's specsmanship game: MTBF and MTTR.

MTBF is mean time between failures. This figure is the statistically average operating time between the start of the disk's lifetime (or its latest repair) and the time of any electronic or mechanical failure. MTBF simply means the average number of hours the disk operates before failure occurs. MTBFs range from 8,000 hours to 20,000 hours.

MTTR is the mean time to repair, the average time needed to repair any type of failure. Rated in hours and minutes, the average MTTR for most hard disk drives is one-half hour.

This specsmanship game is vicious. MTBF is frequently derived from testing some production (or prototype) units under extreme conditions (running the unit in above-average temperatures and on a vibrating surface, for instance). Statistical extrapolations are made from these tests. There is little motivation for a company to revise its MTBF up or—more important—down. MTBF can be a decent indicator of quality but not an exact method of pinpointing disk drive failures.

MTTR is a good indication of how long the repair technician will take to fix the disk drive. However, where is the repair technician located? If the electronics are at fault, a qualified technician can repair the failure at any site. If the technician is not at a local office, the disk drive must be shipped to the technician. If the sealed unit of a Winchester disk is involved, the unit can be repaired only in a clean-room environment. Because these environments are extremely expensive to establish and maintain, you can assume that only the disk drive manufacturer can make this type of repair and the disk drive must be shipped to the factory.

MTTR does not include the time required to get the drive to a repair site. MTTR specifies how long the technician takes to make the repair after the drive is on the repair bench. I have known of disk drives with a "real" MTTR of over three weeks: one day of overnight air shipment, two and one-half weeks waiting for a harried technician to start repairs, one hour to repair, two days of testing, one day of waiting for shipping to package the disk drive, and one more day in overnight transit.

Choosing a reliable disk drive is important. However, most disk drives have similar MTBFs, making this statistic meaningless. MTTR is not a reliable statistic either. Disk downtime includes the shipping and waiting time, which is not covered in the statistic.

Even more important than choosing a reliable disk drive is choosing a reliable service and repair policy. Who will repair the disk drive and where? Who is responsible for the cost of shipping, if needed? How long will the repair take; that is, how long before the technicians can get to the disk drive? If the drive is under warranty, will the vendor simply swap a new disk drive for the failed drive or must the disk drive be sent to the factory for repairs?

The subject of repair is broad. This chapter only introduces the subject. The answers to these questions can decide who will receive your order. Ask questions about service and repair before you buy.

Evaluating the Issues

At this point, this book has covered many key issues of hard disks. Regarding internal versus external hard disk, the general guideline is to use an internal hard disk if possible. Space and power considerations may force PC owners and some XT owners to use an external hard disk instead. Certain types of hard disks come only as external units.

The meaningful statistics—storage capacity, access time, and price—have been covered. The higher capacity or faster disk drive costs more. The best guideline is to choose capacity over speed, but access speeds of 40 ms and under can be essential with the AT.

The discussion has covered some prominent features. Having a bootable hard disk is preferred over starting DOS from a floppy diskette. Automatic-parking disk drives are important for portable computers; people using a desktop computer can easily run the proper shutdown program. Disk drives using Whitney recording heads and thin-film media have better resistance to shock. Forgetting some of the parts you need can be a major annoyance on the day the disk drive is installed.

Backing Up—No Question at All

One final part of the purchase issue must be addressed: how to back up the hard disk. This section covers the basics of the issue. Backup

is presented again in Chapters 15 and 16 with specific examples of ways to handle this important process.

Backups are extra copies of files made routinely and used when something "undesirably alters" the original copy. The subject of backups is similar to the weather: everyone talks about it but many people do little about it. That kind of attitude is fatal.

Why back up? Accidents and other failures happen too regularly. Although regular backups are a relatively painless insurance policy, many people do not bother with regular backups. They think that a regular daily or monthly backup routine is too inconvenient. The chance of losing a file is small and comes too infrequently. The "never happens to me" syndrome sets in. In the wings, Murphy smiles. Disaster comes in many different guises. You can find ample ways to send a file to never-never land, and the "small" odds of losing a file (or the complete hard disk) increase daily.

Even on a floppy disk drive computer, backups are certainly important. Inadvertently destroying the only good copy of a file, such as a word-processing document or spreadsheet, is all too easy. A slip of the fingers and a blank worksheet is saved over the one that contains an in-depth financial analysis. A moment of carelessness causes the erasure of the PLAN5 document when you meant to erase PLAN4B. The faithful floppy diskette used all year to store next year's budgets wears out, and the data cannot be retrieved. The horror stories are numerous.

On a hard disk system, backup is even more important. If a floppy disk drive fails, you do lose data, but you lose only a dozen or so files on the diskette. The time to recover them may take a few hours or a few days. Multiply that potential loss by 20 or 50 times, the conservative equivalent amount of data the hard disk can hold. Any failure of the hard disk can boost the recovery time from hours to weeks. A wrong DOS command can wipe out the information as surely as any head crash.

A major reason many people do not bother with backups is their perception of the inconvenience of making backups every day or every week just to protect against something that doesn't happen that often—especially to "me." These things never happen to "you" or "me," do they? The point is, though, that disaster can come in many different guises. There are enough different ways to send a file to never-never land to raise odds that something will mess up a valuable file for you.

A great deal can be said for the joy of whipping out a backup copy and saving 5 hours (or 25 hours) of redoing a file. On the

other hand, less than nothing can be said for the agony of reconstructing or reentering a group of files that have been trashed. If you provide for and set up a reasonable routine, the task of making periodic backups can be made less onerous. Why wait for the horse to disappear before you close the barn door?

Put simply, the question is not *whether* to make backup copies, but *how* and *when*. Furthermore, if you are planning to buy a hard disk drive, you should be thinking about your backup plans at the same time. The purpose of this chapter is to introduce the subject early because of its extreme importance. Chapters 15 and 16 present in more detail the ways and means of backup.

The Floppy Diskette as the Backup

You have two basic choices in backing up a hard disk: use the floppy diskette or use some special media. The floppy disk drive comes with your computer; the cost of the drive is not additional. Special media need a special device and its cost is additional. Because the floppy disk drive is available, you should consider using it as the backup device.

Software-based Backup

Hardware is half of the solution; the other half is software. You need software to do the backup.

COPY

DOS's COPY command offers one solution. Is there any reason why COPY should not be used to back up at the end of each day any working files that have been changed? Well, yes and no. Yes, you can use COPY easily if (a) you don't have too many files involved; (b) no file is bigger than your floppy's total capacity (362,000, 720,000, or 1.2 million bytes, depending on the particular floppy drives you have); (c) the files to back up are not located in many different subdirectories.

COPY takes too long to back up a large number of files. This fact may deter you from being conscientious about backing up regularly. If the files are larger than your floppy diskette's capacity, using COPY is a physical impossibility. You may be able to split some large files into manageable chunks, but the additional work is another deterrent. If files are located in various subdirectories on the hard disk, careful surveillance and control are required to ensure that you copy them

all. Again, too much time and attention are required for what should be a routine task.

This information is not meant to be discouraging, just realistic. If the process is unwieldy or takes an unreasonably long time, most people will start avoiding the task. If you are working on (modifying or creating) only a selected few files at a time, COPY may fit the bill. Using the IBM AT's high-density (1.2 megabyte) drives also helps.

You can get public domain or shareware COPY-like programs that are closely related to the DOS COPY command. Basically variations on COPY, the programs add some additional features that make backing up more convenient. I will discuss two of these programs in Chapters 10 and 16.

BACKUP and RESTORE

If using COPY (or even the thought of using it) gives you headaches, you need to look for alternatives. The BACKUP and RESTORE utilities were added to DOS V2.0 and are the mainstay of managing hard disk backup with DOS. BACKUP is the file backup utility; RESTORE restores files that have been backed up with this utility. These sophisticated copying commands add features to make backing up easier.

When you use BACKUP, you can specify that you want to back up (1) only files modified or created since the last time you used BACKUP or after the date you specify or (2) files located in any subdirectories of the directory you specify in the command. As a result, with one command (without specifying each subdirectory), you can back up from your hard disk every file that was not backed up the last time you ran the BACKUP utility.

BACKUP can handle files of any size; it simply spills them over as many disks as necessary. In general, BACKUP uses all the disk space needed until the system gets to the last file that can fit on the backup diskette. Then the utility splits that file so that part gets stored on one diskette and the rest on the next diskette in the sequence.

Because of the way BACKUP stores files (unlike the standard COPY command), the only way to bring back those files is to use the RESTORE command. RESTORE is the opposite of BACKUP and offers a couple of convenient options. You can choose to restore files in all subdirectory levels below the directory you specify. You also can choose to be prompted on whether to restore files that are read-only or have been changed since the last backup. This last feature is handy

to prevent you from restoring an older version of a valuable file *over* a version that has been modified since you made the backup.

Although the BACKUP and RESTORE combination is pretty reliable (especially if you turn on DOS's verification option), the major drawback for many people seems to be the speed, or the lack thereof (especially if you turn on verification). You wait for the commands to write the information to the floppies, and then you must add the time to shuffle the floppy diskettes in and out of the disk drives. The number of floppies shuffled is not trivial. Backing up a 10-megabyte drive requires about 27 double-sided floppies, 9 high-capacity (HC) floppies, or 14 microfloppies (the new 3 1/2-inch diskettes). On the AT, 17 HC diskettes are required for a 20-megabyte hard disk, 25 HC diskettes for the 30-megabyte drive. Obviously, larger disk drives increase the number of floppies needed.

BACKUP and RESTORE Cousins

Just as COPY has its clones, so do the BACKUP and RESTORE utilities. A number of software manufacturers have brought out products that function like these utilities and even add convenience features that are missing from BACKUP and RESTORE. Some programs offer terrific speed improvements, make selecting unrelated files easier, permit the use of unformatted diskettes (which saves time), or make "working copies" that can be used without using RESTORE. The advantages of these programs are real: greater ease of use and faster operations.

These programs, including BACKUP and RESTORE, are explored in later chapters. Keep in mind that these programs are software-based; no additional hardware is required. Also remember that each program still uses floppy diskettes for backup, albeit faster and more conveniently and at significantly lower cost than the more technically advanced hardware alternatives.

XCOPY, the Extra Copy Command

XCOPY, a command new to DOS 3.2, addresses some of the drawbacks of using COPY and BACKUP and RESTORE. Like BACKUP and RESTORE, XCOPY is a disk-based external command, meaning that the program resides on a disk and not in the computer's RAM, as does COPY. XCOPY combines features found in both COPY and BACKUP; in certain ways, XCOPY is a cross between the two programs.

You can use XCOPY to copy files from every subdirectory below a declared subdirectory, for example. You also can copy only files

that have been created or changed since the last backup or use of XCOPY. You can instruct XCOPY to prompt you before each file is copied. XCOPY also works faster than COPY by using all your machine's available RAM rather than loading one file at a time as COPY does. These features allow you to back up selected files anywhere on the hard disk, more conveniently than you can with BACKUP.

The major difference between XCOPY and BACKUP is that XCOPY stops when the copied-to diskette fills, while BACKUP continues. Hence, XCOPY cannot handle files larger than a floppy. Files copied with XCOPY can be used directly; BACKUPed files cannot. Files processed by BACKUP must be retrieved by RESTORE before the files can be used. COPY or XCOPY can be used to retrieve the files processed by XCOPY.

Hardware Alternatives

Another backup alternative is to add a hardware device. This choice is undeniably a more expensive solution but can bring its own advantages. A hardware alternative also can influence your hard disk purchase. Some manufacturers place the backup unit in an external cabinet with a hard disk. Some tape units can be mounted in the computer itself. Other manufacturers have removable cartridges that function like a floppy disk. Therefore, an overview of these devices is included in this chapter.

One simple alternative is to go out and pick up another hard disk. Not long ago this suggestion was completely frivolous. With the precipitous drop in hard disk prices, the suggestion has a whiff of sensibility. The odds of two hard disks simultaneously going up in smoke are fairly small. Personally, I don't like the idea of putting all my eggs in one technology; I want to hedge my bets a bit.

The more commonly considered hardware alternative is a tape backup system. Perhaps more interest is generated in this alternative because the prices of tape backup systems have plummeted, and tape backup systems have improved their method of backing up.

Previously, only *mirror-image* tape backups were available. An exact binary image of the files on the hard disk was stored on tape. To retrieve just one file, you first backed up the entire hard disk, retrieved the old copy of the file, copied the file to another media (like a floppy diskette), and then restored the latest copy of the hard disk. The restoration process was less than convenient. The recent trend is to provide also, or instead, a *file-by-file* backup, identical to

the results of BACKUP and RESTORE. Each file is stored individually by name. Restoring the single file has become easy again.

An interesting development in easy-to-use software is background-resident software that, once installed, generates *automatic* backup files with no additional intervention by the user. The software charges into action at prescribed time intervals or at selectable times of day and also waits until you have stopped working at your machine. Although the backups in this automatic mode may take longer than if you are doing them manually, the advantage of complete, automatic backup outweighs the slower speed.

Other than price, the major positive movement for purchasing tape backup units has been the creation of better software. To a large extent, a tape system's ease of use is dependent on the software.

Tapes come in various sizes, from open-reel 12-inch tape to 1/2-inch or 1/4-inch cartridges. The 1/4-inch tape cartridge is the most popular form; the cartridge, commonly known as the TC-200, holds about 10 megabytes of information. Because the cartridges hold much more information than even the 1.2 megabyte high-capacity floppy diskette, many people purchase tape systems solely to avoid backing up with a "floppy shuffle."

A hardware alternative that appears to compete directly with tape drives but really needs to be considered in a larger context is the removable cartridge drive, the most famous of which is probably the Bernoulli Box.

These units are undoubtedly in the "high-priced spread," but the counterargument is that they are much more than just backup systems. The speed, convenience, and capacity of the removable cartridge drives permit them also to function as primary storage devices and so to compete directly with hard disks on their own "turf." You have to consider both the primary and backup storage capacities in your decision.

Another hardware solution uses a device that has become quite common today, the video cassette recorder. Some clever firms have developed an interface controller card that translates the data produced by a computer into a form that can be laid down and stored on videotape. One $5.99 videotape can store up to 80 megabytes! You can buy a specially equipped VCR that adds more automated features, or you can simply hook up any old VCR that you already own.

The ultimate in primary and backup storage, however, may be down the road yet. Optical disks that resemble audio CDs offer a

great deal of promise and a gigantic amount of storage capacity. In 1986 and 1987, we are beginning to see the so-called Write Once, Read Mostly disks and players, which would at least allow for long-term archival backup of information that doesn't change but needs to be added to. Any prediction is difficult, but these devices may become the major source of backup for the next generation of computers.

Summary

This chapter has introduced many terms and concepts about hard disks and given some guidelines for selecting a hard disk. The chapter has talked about the important process of backup and given a brief overview on the hardware and software alternatives.

The important points to remember are the following:

- If you have sufficient space and power, the preferred choice is the internal hard disk. An external unit can be chosen if space and power are insufficient or a larger capacity unit or an integral backup unit is desired.
- Storage capacity and access time are reflected in a hard disk's price. Choose capacity over speed, but choose access times of less than 40 ms for 286- and 386-based systems like the IBM Personal Computer AT.
- Prefer hard disks with features like automatic parking, the ability to start DOS from the hard disk, smaller heads, and hardened media.
- Examine the service and backup issues before selecting the hard disk.

The next chapter discusses how to set up your chosen disk.

Chapter 4

Setting Up the Hard Disk

This chapter presents detailed instructions for installing and starting a hard disk in an IBM microcomputer. The chapter is divided into two sections, which deal with physical installation and logical installation.

Physical setup covers the steps in placing the hard disk into the computer and making the correct electronic connections. Logical setup covers the steps which prepare the disk for use by DOS.

If you already have a hard disk in your computer, you might browse through the first section and read the second section. If your hard disk is already up and running, you might skim the latter part of the chapter about configuring your system and skip to the next chapter. If you're just starting out with a new hard disk, reading this entire chapter will benefit you.

Physical Installation

With hard disk prices falling rapidly and capacities increasing, many of you are considering upgrading your disk storage. The lower prices are tempting, but reading ads for hard disks presents one problem. No one advertises "installed" prices, and few mail-order firms will do the installation for you.

Putting a hard disk into a PC XT or AT isn't really difficult. You don't solder any components; you may not need to touch even a single chip. If you feel comfortable with a screwdriver (and with staring into the electronic innards of your PC), you can do your own installation.

Because you are going to be poking around a reasonably complex piece of electronic equipment, you should be aware of the dangers of static electricity. The danger is not to you but to the equipment. Much of the circuitry of a computer is rated to work at a range of 5 to 15 volts. The spark you feel after crossing a carpet is between 4,000 and 10,000 volts. Because a voltage of more than 30 is fatal to your computer's chips, you must deal with the problem of static electricity.

First, test your working environment. If you see, hear, or feel sparks when you walk through the room (and it happens for reasons other than your sparkling personality), you should think twice before charging ahead.

I have heard of people who ground themselves by tying one end of a long wire around one wrist and the other end around a radiator or water pipe before working on computer equipment. I'm not suggesting that you need to go to this much trouble; I have never found it necessary myself. Step 2 in the following discussion explains a simpler way to discharge any static electricity.

Obviously, I can't describe the exact details of every different hard disk installation. The following steps are 90 percent complete and cover the essentials of most hard disk installations.[1] The major steps can be summarized as follows:

1. Turn off the computer, unhook any cables plugged into the rear connectors of the unit, and remove the cover.
2. Touch the outside of the power supply to discharge static electricity and then unplug the power cord.
3. Make room for the hard disk drive by disconnecting and removing a floppy disk drive or a hard disk drive, or by removing a faceplate.
4. Free the appropriate expansion slot by shifting around some cards if necessary.

---

[1] *Much of the information concerning the detailed steps of physical installation was furnished by Seagate Technology, 920 Disc Drive, Scotts Valley, CA 95066.*

5. Connect the cables to the disk controller card and set its switches or jumpers if necessary.
6. Install the board in its expansion slot.
7. Change the computer's configuration by modifying its internal switch settings.
8. Adjust the hard disk, slide it into the system unit, and connect the cables from the controller card and the power supply.
9. Add a faceplate, if needed to fill an empty space on the front panel.
10. Secure the hard disk drive, arrange the cables neatly in your system unit, and replace the system unit cover.

Now, look at each step in more detail.

Step 1. Turn off the computer, unhook any cables plugged into the rear connectors of the unit, and remove the cover.

First, turn off the computer. This step seems natural. And forgetting this step is a health hazard—to yourself and your computer equipment. Do not, however, disconnect the power cord. Maintain a healthy respect for the connectors from the power supply to the system board until you've completed Step 2.

Next, disconnect from the back of the computer any cables to external devices. If you take a minimalist view, just disconnect any cable that interferes with pulling off the cover. If you're unfamiliar with the cables, label each cable and receptacle to help you reconstruct the jigsaw puzzle later.

To remove the system unit's cover, remove the cover's mounting screws located on the rear of the unit. The early PCs used two screws; later PCs and all XTs and ATs use five screws (see figs. 4.1, 4.2 and 4.3).

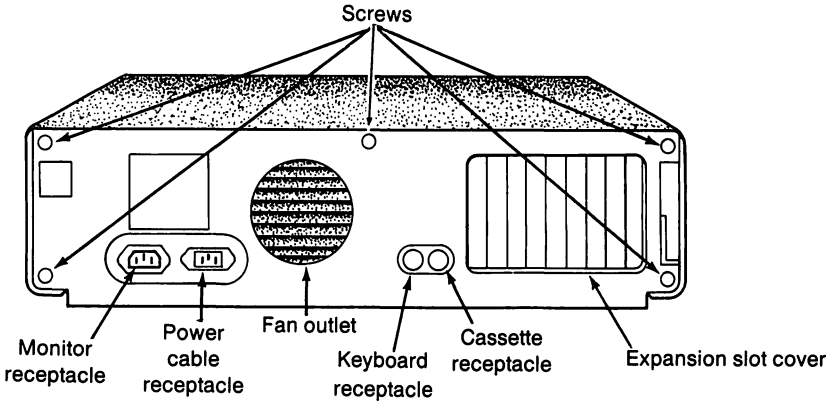


Fig. 4.1. *The exterior back of the IBM Personal Computer.*

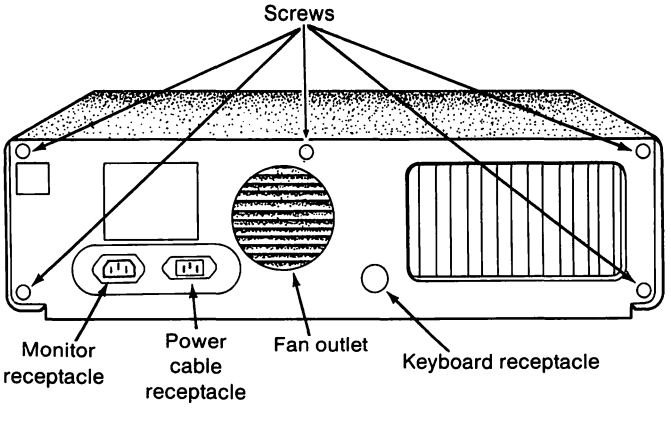


Fig. 4.2. *The exterior back of the Personal Computer XT.*

On the AT, you may have two additional steps. If the brownish-colored back vanity plate is installed, it must be removed. Just pull the plate off its velcro fasteners. Also make sure the keylock on the front panel is in the unlocked position. If not, a small metal bar attached to the keylock prevents you from removing the cover.

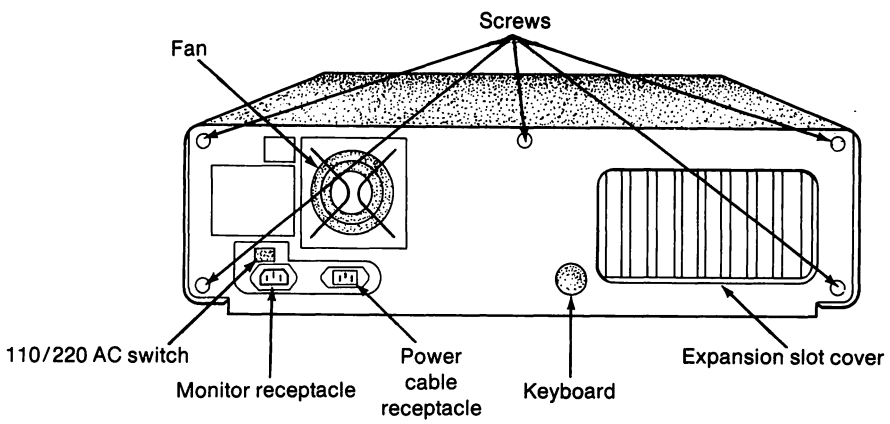


Fig. 4.3 *The exterior back of the Personal Computer AT.*

After you remove the screws (and the AT's back panel), slide the cover toward the front of the computer until the cover stops. Then tilt up the front edge of the cover (see figs. 4.4 and 4.5); it should come off from the system unit.

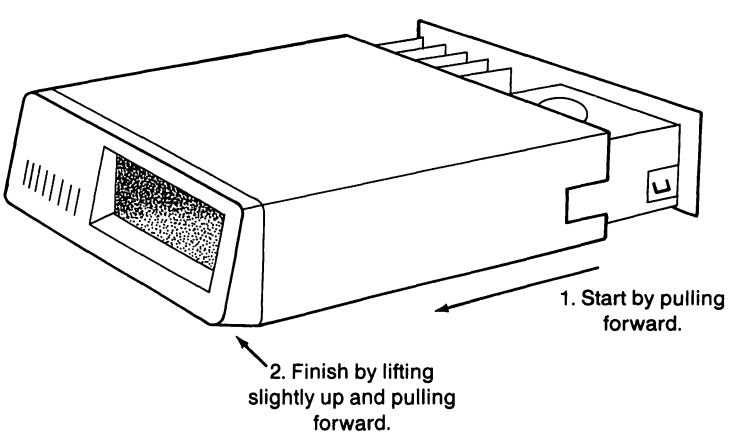


Fig. 4.4. *Removing the cover of the PC or XT.*

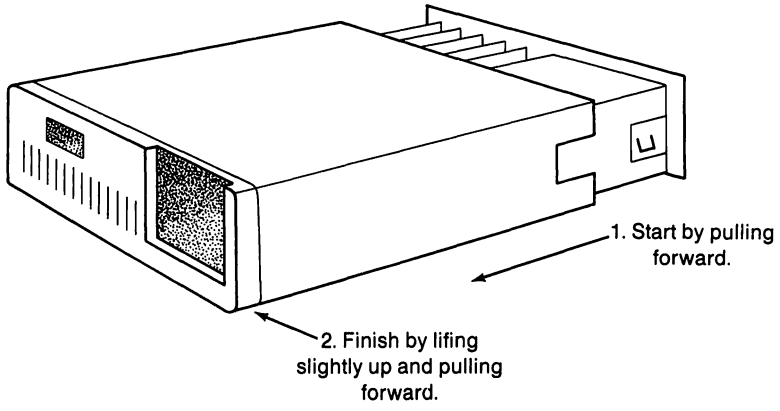


Fig. 4.5. *Removing the cover of the AT.*

Step 2. Touch the outside of the power supply to discharge static electricity and then unplug the power cord.

While the power cord is still attached, the outside of the power supply is connected to earth ground. Touching the top of the power supply discharges any static electricity you may be carrying (see figs. 4.6 and 4.7). While touching the top of the supply, also touch each tool you will use and the hard disk itself. This action "equalizes" the units (discharges any static electricity).

Because you will be working inside an electronic device, unplug the power cord to ensure that a stray move will not present a shocking surprise.

Step 3. Make room for the hard disk drive by disconnecting and removing a floppy disk drive or a hard disk drive, or by removing a faceplate.

If the bay where you will install the hard disk is covered by a faceplate, you need to remove only the faceplate because nothing else is behind the plate. The faceplate is usually anchored to the front of the computer by a couple of metal clips. Turn the screws and remove the clips.

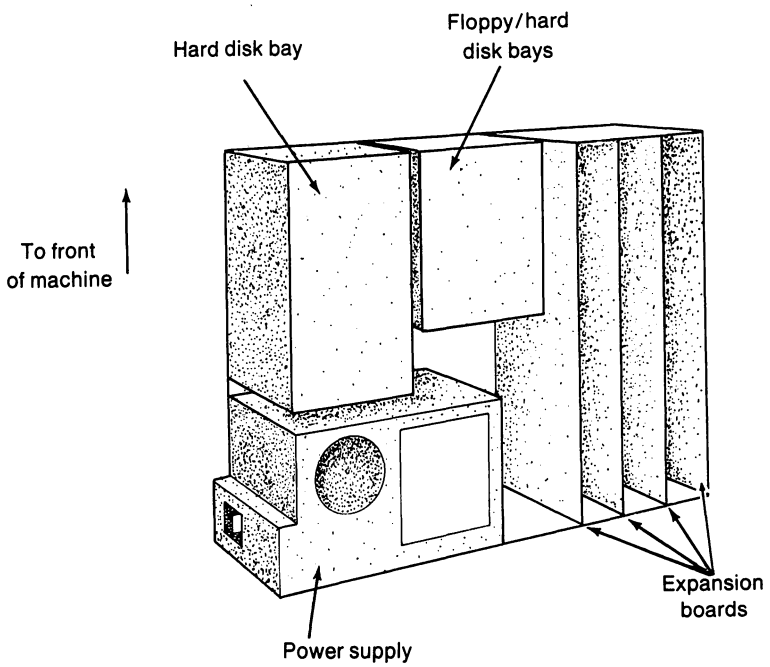


Fig. 4.6. *The interior of the PC XT.*

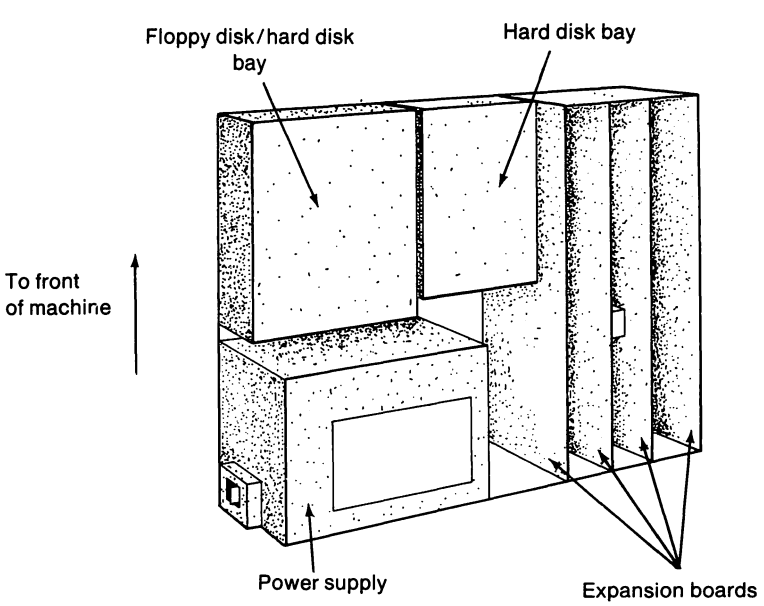


Fig. 4.7. *The interior of the Personal Computer AT.*

If you are substituting a hard disk drive for a floppy disk drive, you must remove the floppy disk drive. It is held in place by two or three screws mounted either in the side and bottom (most PCs and XTs) or front of the unit (ATs). Remove the screws and gently slide the floppy disk drive forward just enough that you can lay your fingers on the cables plugged into the rear of the drive (see figs. 4.8 and 4.9). One flat ribbon cable connects the drive to the controller card. A second multiwired cable connects the drive to the power supply. Carefully disconnect both cables. Grasp the connectors and gently pull, wriggling the cables sideways if needed. Then slide the drive all the way out of the front of the system unit.

When removing or inserting any screws, be careful that the screws don't fall into the computer. A metal screw left inside the computer can cause a short circuit and damage the system. Fishing for a lost screw in the computer's innards is both time-consuming and frustrating. More than one person has had to disassemble the entire computer to recover a lost screw.

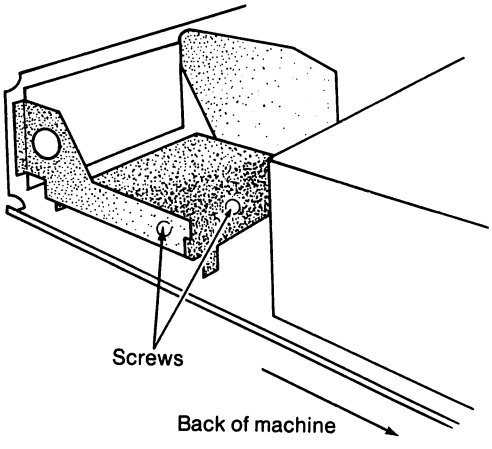


Fig. 4.8. *Side view of the PC and PC XT showing the bay for the hard disk.*

Step 4. Free the appropriate expansion slot by shifting around some cards if necessary.

Follow this step if you are working on a PC or XT or if you are using a nonfixed disk on an AT.

Usually, the hard disk controller card must be installed in the expansion slot immediately to the left of the floppy disk drive controller card. This slot is the one to which the floppy disk drives

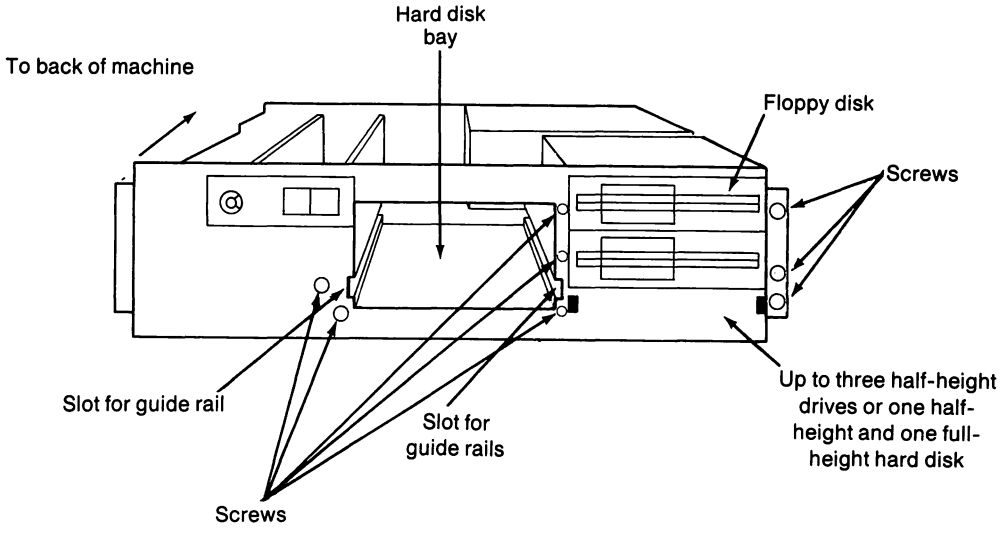


Fig. 4.9. *A front view of the Personal Computer AT exposing the hard disk bay. A bay for a full-height hard disk is located in the center. If a disk drive does not occupy the middle bay on the right, a gray face plate covers the bay (not shown). If a full-height drive is not used, a black metal face plate covers the lower bay.*

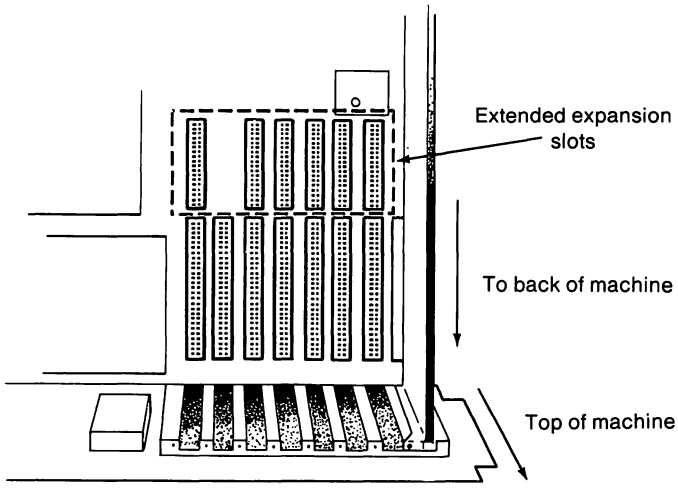


Fig. 4.10. *View from above of the AT's expansion slots with right slot occupied. The eight slots use the same 50-pin connectors as the PC and PC XT. Six of the slots also use an extended 34-pin connector.*

are connected (see fig. 4.10). Any board in the slot to the left of the floppy disk card must be relocated.

First, identify a vacant expansion slot for the board you want to move. Then remove the slot-cover screw from that slot and lift out the slot cover.

Second, remove the board you want to relocate. Grasp the board on both sides with both hands and gently lift up. A slight rocking motion helps dislodge a stuck board.

If the plastic guide at the front wall of the expansion slot is missing, install a new guide (see fig. 4.11). Gently slip the board into its new slot and secure the board by replacing the slot-cover screw.

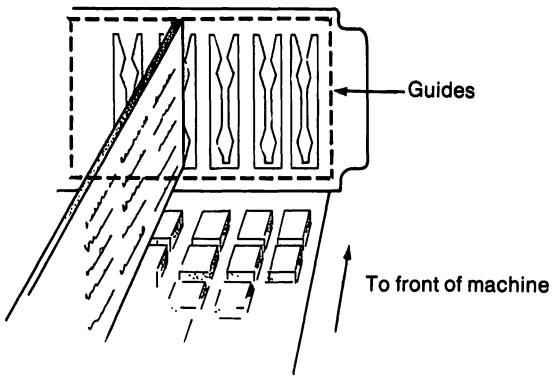


Fig. 4.11. *Plastic guides, on the front inside wall of the computer, that secure full-length expansion boards. Different guides may be used in different models.*

Step 5. Connect the cables to the disk controller card and set its switches or jumpers if necessary.

Most fixed disks use two flat ribbon cables: a smaller controller cable and a wider data cable. If the cables are not already installed on the controller card, plug the two cables into the card. If a hard disk is already installed, you may need to connect only the single controller cable. The installed data cable has two connectors for two disk drives.

Most controller cards use a ribbon cable with a dark red or blue stripe at one end of the cable (see fig. 4.12). When you attach the cable, the connector should be turned so that the dark stripe is down (PC or XT) or toward the back of the unit (AT).

The documentation accompanying your hard disk and controller tells whether certain switches and jumpers need to be set. Most switches have a legend that reads "ON" or "1." Sliding the switch in that direction turns on the switch; moving the switch the other way

turns it off. You can use a pen, pencil, or small screwdriver to slide the switches to the proper settings.

A jumper is a small insulated connector that "jumps," or connects, two contact pins on the board. To change the jumper, use a small screwdriver gently to pry off the jumper, or pull off the jumper with needle-nose pliers. You can install the jumper on the proper pins with your fingers.

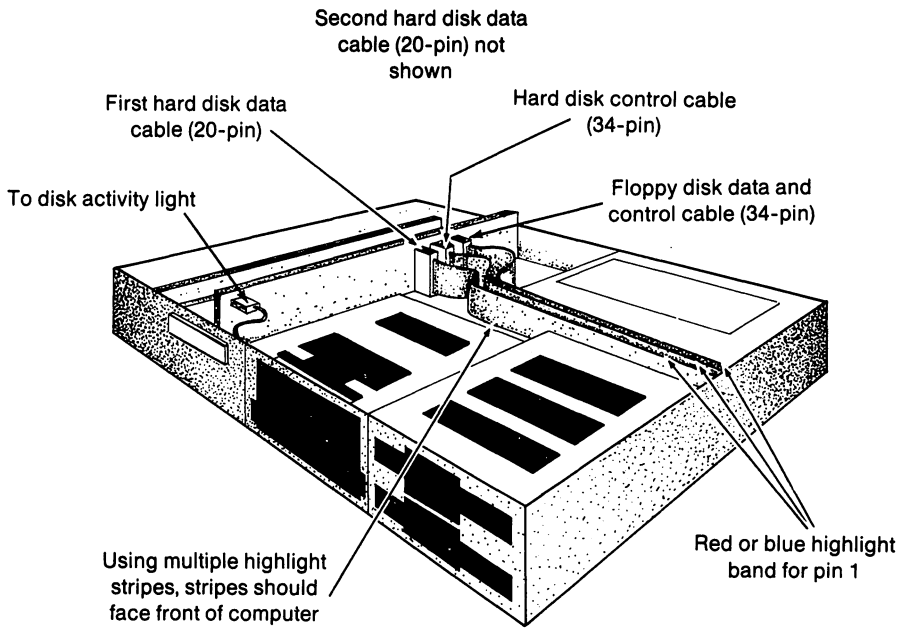


Fig. 4.12. *Sketched is the fixed-floppy disk adapter of the Personal Compter AT. Note that the cables, whose connectors can vary in shape, should be mounted with the highlight stripe for pin 1 at the top of the cable. If the cable uses highlight stripes approximately every half inch instead, the side with the stripes should face the front of the machine.*

Step 6. Install the board in its expansion slot.

If the plastic guide is missing from the front plate of the expansion slot, install a new guide.

Gently slip the controller card into the expansion slot, using a rocking motion if needed, until the card fits firmly in the connector on the motherboard. Snake the cables carefully around the right side of the floppy disk controller card and the back of the floppy disk drive, if the machine has one. Use the slot-cover screw to secure the board (see fig. 4.13).

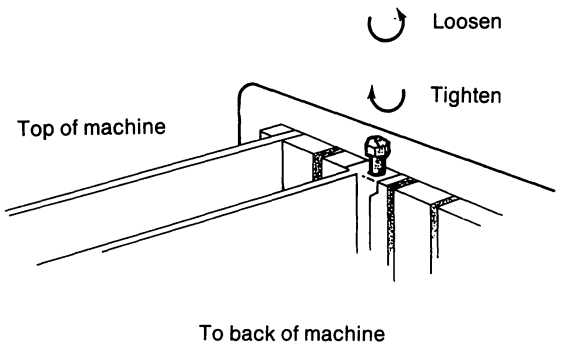


Fig. 4.13. *Each expansion board or expansion slot cover plate is secured by a single screw.*

Step 7. Change the computer's configuration by modifying its internal switch settings.

This step is for PC and XT owners who are replacing a floppy disk drive with a hard disk.

The configuration switches on the motherboard must be changed. Consult your computer's operating manual for the exact settings. Using a pencil, pen, or small screwdriver, change the switches as directed.

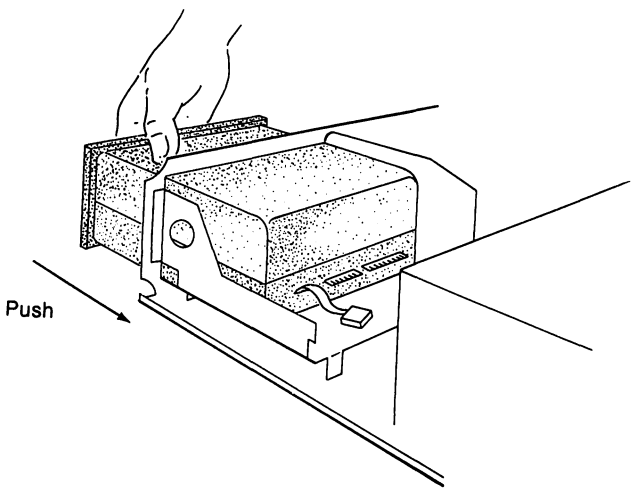
Step 8. Adjust the hard disk, slide it into the system unit, and connect the cables from the controller card and the power supply.

Some disk drives have jumpers, "terminating" resister packages, that need to be moved or removed. To make the needed changes, follow the directions accompanying the drive. Pay careful attention to these instructions if this is a second hard disk drive; adjustments must be made on the disk drive to indicate that it is the second hard disk.

If you are installing the disk drive in an AT, make sure the guide rails are fastened to the disk drive. Place the hard disk drive in the opening in the front of the system unit. Slide the drive about three-quarters of the way into the unit, leaving enough room to attach the cables to the disk drive (see fig. 4.14).

Connect to the drive the two cables from the controller card and one cable from the power supply (in the right rear of the system unit). If the connectors from the controller card are not keyed to fit only one way, the connector usually has a red stripe on the side. Align this stripe with the slot in the card edge. If you are using

cables already in the unit, IBM letters each connector that fits on the disk drive. The lettered side faces up. On the AT, you also have a ground wire using a flat metal slot connector. The ground wire is attached to a fitting post on the disk drive.



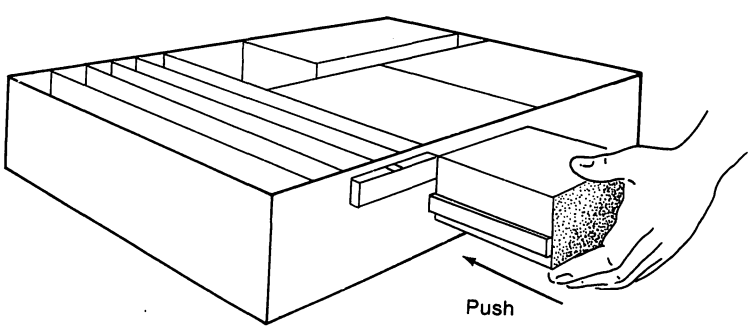


Fig. 4.14. *A hard disk being mounted in a PC XT (above) and into the bay at the center of the PC AT (below).*

Step 9. Add a faceplate, if needed to fill an empty space on the front panel.

This step applies if you are replacing a full-height disk drive with a half-height disk drive. The faceplate usually just snaps over the front of the empty gap.

Step 10. Secure the hard disk drive, arrange the cables neatly in your system unit, and replace the system unit cover.

Finish sliding the drive all the way into the system unit. Secure the drive with one or two side or front screws (depending on the computer), and arrange the three cables behind the hard disk drive so that they do not interfere with anything else.

Replace the cover by tilting it slightly upward and slipping it on to the guides on the system unit. Move the cover so that it is parallel to the floor and slide back the cover until it stops. Don't attach the cover until you have tested the system (you may have to dig back in there again).

"High Tech" Tips

Here's a recommended "high-tech" modification for most PCs or XTs. Using electrical tape, cover the ventilation slots just below the left and right disk housings. Leave an opening about six inches long on the left part of the ventilation slot. The purpose is to enhance the air flow around the expansion cards on the left side of the computer, where ventilation is most needed.

When remounting cables, it is easy to forget which cable mounts to what adapter or peripheral. The task of remounting cables is easier because most cables have a shape "memory." Bends and folds in a cable come from the cable's position in the computer over a period of time. To rehook the cables, restring the cables as best you can remember. Then look at the cable's bends, folds, or curves. Often the shape of the cable will show you how the cables should be threaded in the system, which cable goes to which connector, and which end of the connector is "up."

Differences in Installing an External Drive

An external hard disk drive comes in its own separate housing but requires a connection to the computer. Generally, you don't

install the hard disk; you install only its controller board. One or two cables from the back of the controller card connect to the external unit.

Follow the preceding directions for installing the controller card. Then attach the cables from the back of the card to the hard disk. The cables normally attach to a connector on the exterior of the slot cover. No cable runs from within the computer. Because the external hard disk has its own power supply, no cable connects the computer's power supply with the hard disk.

Differences in Installing a Hard-Card

Of the disk drives mentioned earlier, the hard-card (Winchester disk on an adapter card) is the easiest to install. The whole installation process from first picking up the screwdriver to replacing the cover usually takes less than 10 minutes, slightly more if some expansion cards must be rearranged to make room.

The hard-card has no cables to connect. The card gets its power through the expansion slots of the motherboard, so no power cable is needed. Because the hard disk is an integral part of the controller board, the controller and data cables are part of the board's circuitry. In essence, the hard-card is just plugged into a slot like any other expansion board.

The major difference is size. Does the card fit within the dimensions of a single slot, or does the card take up one-half to one more slot? If the hard-card is truly one slot big, any expansion slot should work. Plus™ Developments HardCard is an example of a hard-card that takes only one slot.

If the card laps over the next expansion slot, you need two adjacent slots. You may need to move one or two expansion cards to free two adjacent slots.

The "middle ground" is the slot-and-one-half hard-card. This hard-card needs the forward half of a second slot. A half-sized card, such as the IBM serial or parallel adapters, can be used in the second slot. If you have only full-length cards in your machine, you still forfeit an additional free slot to use the hard-card. This limitation has some exceptions. For example, the Mountain® DriveCard can fit into the farthest left slot of an IBM PC, but only if you move the computer's speaker out of the way. (This procedure is easier than it sounds.)

To install the hard-card, follow the instructions for installing any controller card. Remember that thicker hard-cards may require you to shuffle some expansion boards.

Some Hints for the Experienced or Wary

Experienced technicians know that if anything can go wrong, it will. For this reason, you might vary the installation process slightly. Don't screw anything in and don't replace the cover. Turn on the computer and see whether the disk drive works. This method is jokingly called the "smoke test." Any smoke means that the computer has failed the test and something is wrong. Personally, I've never seen smoke from a disk drive, but I have heard funny noises or noticed that the red light on the disk drive has failed to light. If anything seems unusual, turn off the power immediately and check all connections and settings.

After you see that the computer properly recognizes the disk drive, turn off the system and secure the expansion cards and disk drive. Turn on the system again, test the drive, turn off the system, and replace the system cover.

As a result of personal experience, I format the hard disk, then "exercise" the disk drive for about twenty minutes before putting any screws back in. I have too often found that I forgot to set a switch or that the manufacturer's instructions were vague. By seeing the disk drive work, I know the proper connections were made and the drive can be safely mounted.

The only caution for this process is not to hit or move the computer. The drive and controller card are not secured, and either could move and short itself or some other part of the computer.

If you are wary of these instructions and feel uncomfortable about installing your own hard disk, the easiest (although not cheapest) solution is have a local dealer install the unit. The process leaves your pocketbook lighter but frees you from any headaches of installing and testing the hard disk.

Logical Installation

Although the hard disk has been placed in the computer and all the appropriate cables are connected, the drive is not ready for use yet. As mentioned in Chapter 2, the disk must be prepared. A "raw" hard disk needs three preparation steps, unlike the two preparation steps for a floppy diskette. Assuming no disasters or major changes, two of the steps are performed only once during the life of the hard disk.

The three steps are absolute, or physical, formatting of the disk; partitioning the disk; and relative or logical formatting. Partitioning

the hard disk is the step not mentioned in Chapter 2. This subject is discussed in a following section.

Your hard disk must be physically, or absolutely, formatted before you take the steps given in the following sections. Because the physical formatting is usually performed before you get the disk drive and physical formatting programs vary widely, this book does not cover this step. If absolute formatting has not been performed on your non-IBM disk drive, follow the instructions accompanying the formatting program.

After the hard disk has been physically formatted but before it is logically formatted, it must be partitioned. The statement "creating a DOS partition" implies that different types of partitions can be made or added. That assumption is correct. IBM was very flexible in setting up the hard disk. DOS, or any of several different operating systems, can be used exclusively or with another operating system on the same hard disk. IBM allows you to select the *active* operating system, the operating system in which the computer starts automatically when you turn it on or reboot.

For practical reasons, this discussion is limited to MS-DOS and PC DOS. Certainly, alternative operating systems are available (UNIX and PICK, for example), but space does not permit the discussion of other operating systems. The important fact is that the hard disk can be divided to accommodate several different operating systems.

An Important Note

The following steps assume that your hard disk is new and that your vendor has not installed useful programs on the disk yet. Some dealers perform the work of partitioning and formatting the hard disk when you purchase your system. The dealer also may have installed the applications programs you have purchased on your disk.

If your dealer has installed any useful programs (except the DOS operating system) on your hard disk, *do not perform the instructions in the next three sections!* The steps will destroy any programs on the hard disk.

If you are unsure whether your dealer has done this work, you should contact the dealer for this information. If you cannot contact your dealer, attempt to start your computer from the hard disk (turn the computer on and leave the door to drive A open). If DOS successfully starts, you should not perform the following partitioning and formatting operations.

You may find the sections on establishing your own AUTOEXEC.BAT and CONFIG.SYS files helpful.

Partitioning the Hard Disk for DOS

The examples in this section and the next use PC DOS and IBM fixed disks. The programs reside on the DOS System Master diskette and are called FDISK.EXE and FORMAT.COM. If you have a non-IBM hard disk or have a "work-alike" computer, your software may be different. Check your manufacturer's instructions for partitioning and formatting your hard disk. Most "work-alikes" do just that, so the examples should be fairly close to what you need to do to your system.

This discussion is based on the assumption that you are partitioning the first fixed disk drive. If you are adding a disk drive, complete Steps 1 through 3, and then choose option 5 of the FDISK menu (Select Next Fixed Disk Drive). You'll see a 2 for a fixed disk drive number. The other steps are identical.

AT owners have an extra step to complete before starting this section. You must run the SETUP program, which is located on the *Diagnostics for IBM Personal Computer AT* disk in the *Guide to Operations*.

If FDISK displays a message drastically different from the ones shown here, you have a problem with your hard disk. Check to see that the disk is installed properly. If your hard disk is new, check to see that it has been physically formatted. If the disk installation is correct and the absolute format program has been run but you still get a different message, your hard disk or controller card is defective.

The major steps in the partitioning process are as follows:

1. Insert your DOS System Master diskette in drive A, close the drive door, and start your system.
2. Verify or enter the correct date and time.
3. Start the partitioning program, FDISK.
4. To create a DOS partition, choose option 1 (the default) by pressing the Enter key.
5. Start DOS again.

Here are the steps in detail.

Step 1. Insert your DOS System Master diskette in drive A, close the drive door, and start your system.

You start the system either by turning on the power (if the computer is off) or by pressing the Ctrl-Alt-Del key combination. (Hold down the Ctrl and Alt keys and press the Del key, and then release all three.)

When you start the computer by turning on the power, the system executes a power-up self-test (POST). The self-test can take up to a minute, depending on the type of computer (PC, XT, AT, etc.) and the amount of RAM (random-access memory). If you use the *system reset* or *warm boot* sequence (Ctrl-Alt-Del combination), the computer skips the self-test.

Step 2. Verify or enter the correct date and time.

If your computer has a built-in clock (like the AT), verify the date and time. If the date and time are correct, just press the Enter key after each prompt. If neither is correct (the internal clock is off or the computer does not have a built-in clock), enter the correct date and time at the appropriate prompts. In the date you may use slashes or hyphens as separators (4/10/86 or 4-10-86). Use the colon as the time separator (11:10 or 16:40:20).

Step 3. Start the partitioning program, FDISK.

At this point, you should see an A> (the prompt) on your screen. To start the partitioning program, type

FDISK

Then press the Enter key.

The computer responds with something like the following:

```
IBM Personal Computer
Fixed Disk Setup Program Version 3.10
(C)Copyright IBM Corp. 1983, 1985

 FDISK Options

 Current Fixed Disk Drive: 1
 Choose one of the following

 1. Create DOS Partition
 2. Change Active Partition
 3. Delete DOS Partition
```

```
4. Display Partition Data
5. Select Next Fixed Disk Drive

Enter choice: [1]
```

Step 4. To create a DOS partition, choose option 1 (the default) by pressing the Enter key.

The following message indicates that a DOS partition has already been created on your hard disk:

```
DOS partition already exists.
```

If you see this message, you may safely skip this section and move on to formatting the disk. To exit from FDISK, press the Escape key (Esc). The A> reappears.

If you don't get the preceding message, your screen should look something like this:

```
Create DOS Partition

Current Fixed Disk Drive: 1

Do you wish to use the entire fixed
disk for DOS [Y/N]..............? [Y]
```

Press the Enter key to tell FDISK that you want the entire hard disk to be used by DOS. The red light for the hard disk should go on and off briefly, and the following message should appear:

```
System will now restart

Insert DOS diskette in drive A:
Press any key when ready . . .
```

Step 5. Start DOS again.

The DOS diskette should still be in drive A from Step 1. Press any key to restart DOS. Answer the date and time queries again (as in Step 2). Now you are ready to start formatting the hard disk, as explained in the following section.

As you can see, the process is fairly painless and should take less than two minutes. The next process is almost as simple but takes slightly longer.

Formatting the Hard Disk

The next step is to format the hard disk. You use the DOS FORMAT command, which does several things. The FORMAT command magnetically divides the hard disk into sections and sets up some needed housekeeping areas. You will also direct FORMAT to place important parts of DOS on the hard disk so that the disk is startable (bootable).

The formatting steps are summarized as follows:

1. With the DOS System Master diskette in drive A, start the formatting program.
2. Respond to the query to proceed with the FORMAT command.
3. Provide a volume label.
4. To test that everything is working, start DOS from the hard disk.

Consider these steps in more detail:

Step 1. With the DOS System Master diskette in drive A, start the formatting program.

You start the formatting program by typing

FORMAT C: /S /V

Press Enter.

This command tells DOS to load the FORMAT program from the diskette and use drive C (the hard disk) as the drive to format. If you are formatting the second hard disk, change the **C:** to **D:**. (Remember: if you do this, all the messages will say D: instead of C:.)

The */S* in the command tells DOS to put the *system* files (the key parts of DOS that allow you to start DOS from the disk) on the formatted disk. The */V* allows you to put a *volume label* on the formatted disk. This electronic name is displayed when you request a list of files with the DIR (directory) command or when you use several other DOS commands. The volume label is useful for identifying a particular floppy diskette or hard disk. A volume label is not a necessity but an organizational nicety.

After a few seconds you should see the message:

```
WARNING, ALL DATA ON NON-REMOVABLE DISK
DRIVE C: WILL BE LOST!
Proceed with Format [Y/N]?_
```

Step 2. Respond to the query to proceed with the FORMAT command.

Type **Y** for Yes and press Enter. The following message appears:

```
FORMATTING...
```

This message informs you that DOS FORMAT is doing its magic. The message and the operating light on your hard disk drive will stay on for a few minutes. Then the FORMAT program will signal you with two messages, displayed a few seconds apart:

```
Format complete

System transferred
```

These messages tell you that DOS has completely formatted the hard disk, and certain essential DOS system files have been copied from the floppy diskette to the hard disk. You're almost done.

Step 3. Provide a volume label.

A volume label can be considered a hard disk nickname. In a volume label, you can use only alphanumeric characters, the space, and the following special characters:

```
$ # @ ! & ( ) - {} ' _ ~
```

Because you specified the */V* option to FORMAT, the system now displays the following prompt:

```
Volume label (11 characters, ENTER for none)?_
```

Type a volume name for the disk drive (such as DOS DISK or Don's DISK) and press Enter. FORMAT writes the volume name on the disk, displays a summary of statistics about your hard disk, and drops back to DOS (you see the A>). If you use a 20-megabyte hard disk, you see a message looking something like this:

```
21213184 bytes total disk space
   63488 bytes used by system
21149696 bytes available on disk
```

The number of bytes shown varies with different sizes of hard disks and different versions of DOS. In the case of the 20-megabyte hard disk and DOS V3.1, 63,488 bytes are used by the DOS system files that FORMAT copied to the hard disk.

Step 4. To test that everything is working, start DOS from the hard disk.

Open the floppy disk drive door, do a system reset (Ctrl-Alt-Del), and watch the results. The screen will go blank, the red lights of the floppy and hard disk drives will flash briefly, and you'll hear a beep. The floppy disk drive's red light should turn on for a few seconds; then the red light of the hard disk should come on. After another few seconds, the system should prompt you to verify or enter the current date and time. After you have completed the date and time questions, DOS issues its customary introductory message about the version of DOS used and displays a prompt, C>.

Congratulations, you have successfully installed the hard disk. If you don't get a request for the date and time, reboot the system from your DOS master diskette and try the formatting process again.

The steps for partitioning and formatting the hard disk are fairly mechanical; little critical judgment is involved in the process. In theory, nothing more needs to be done to use your hard disk.

To use the disk *effectively and efficiently*, however, more work is needed. The last start-up step in making your hard disk conveniently usable is to do some simple housekeeping and establish some files that make DOS and your applications programs work faster.

Creating Your First AUTOEXEC.BAT File

A batch file is a file of readable text that contains a series of DOS commands. The advantage of a batch file is that you can execute the series of commands in the file consecutively (in a "batch") just by typing the file name.

Batch files are discussed in detail in Chapters 7 and 8. The subject is introduced here because you ought to use one batch file immediately. You are going to create a simple batch file and later explore other useful things batch files can do.

This first batch file is called AUTOEXEC.BAT. AUTOEXEC is short for *automatic execution*. BAT is the extension for any batch file. When you start DOS (by turning on the computer or doing a system reset), DOS automatically searches for a batch file named AUTOEXEC and runs each command in this batch file. If the file is missing, DOS reverts to asking for the date and time and displaying an introductory copyright message.

How do you get the series of DOS commands into a file? Ordinarily, for a file of some length, you use a text editor or a word processor that can create ordinary text files. However, the AUTOEXEC.BAT file described here is so short that you can employ a shortcut that uses the DOS COPY command.

In brief, the steps for creating the AUTOEXEC.BAT file are

1. Enter the COPY command.
2. Enter the lines of data.
3. Signal DOS that you have finished entering the data.
4. Test the AUTOEXEC batch file by rebooting the system.

Now look at the details of each step.

Step 1. Enter the COPY command.

To use the COPY command, you type

COPY CON: C:AUTOEXEC.BAT

Press the Enter key. Watch where you put spaces; spaces, or lack thereof, are important to DOS.

This command line tells DOS to "copy data, line by line, from the console (CON) input device (the keyboard) and place the data in a file called AUTOEXEC.BAT (the autoexec batch file) on drive C."

Step 2. Enter the lines of data.

If you are using a computer with a built-in clock (the AT, for example), you ought to type some introductory message

Echo Welcome to the IBM AT, Hillary . . .

that will "echo" to the screen when you start up your computer. (You fill in an appropriate first name, of course.) Press the Enter key.

The mere existence of an AUTOEXEC batch file can keep the system from asking for the date and time during each start-up. On the AT this is fine because the AT has an internal clock that keeps track of the date and time. If your computer does not have a built-in clock, you type

date
time

Press Enter after each line. Without these lines, a computer without a built-in clock will not set, or ask for, the correct date and time.

(If you have installed a memory expansion card in your computer, the card may include a clock and the software to set the clock. In this case, you should determine from the expansion card documentation the name of the program needed to activate the clock. That file should be copied to the hard disk and the name of the file should go in the AUTOEXEC.BAT file.)

Step 3. Signal DOS that you have finished entering the data.

Press function key F6 and then press Enter again. These two keystrokes signal DOS that you have finished entering lines of data. The hard disk will churn for a split second, and then you should see the message

```
1 File(s) copied
```

DOS is a great equalist: it sees all devices as equals. The console is just another device like a disk drive or a printer. Therefore, you can store "files," or lines of data from a keyboard, in a disk-based file. The COPY CON technique is a useful quick-and-dirty way of creating any text file, like a batch file, particularly if the amount of text is small. The alternative is to use a text editor (including the line editor EDLIN that comes on the DOS System disk) or a word processor that can edit plain ASCII text files.

Step 4. Test the AUTOEXEC batch file by rebooting the system.

Press the Ctrl-Alt-Del key combination and watch what happens. After DOS "sees" that drive A is empty (or the disk door is open), DOS checks drive B (depending on your system), then goes to the hard disk to start DOS. If your computer has a built-in clock, within a few seconds you should see

```
C> Echo Welcome to the IBM AT, Hillary...
Welcome to the IBM AT, Hillary...
```

If your computer doesn't have a built-in clock, you should see the prompts for date and time. If you don't see either, try retyping the COPY command and the lines for the batch file. Do a directory of C drive by typing **DIR** and look for the file called AUTOEXEC.BAT in the listing.

(Although subdirectories have not been discussed yet, you need to know that the AUTOEXEC.BAT file goes in what is called the root directory. Because you are starting DOS from scratch, the start-up directory on a disk drive *is* the root directory so you have nothing to worry about. If some of these setup steps have already been done, other directories may have been created.)

This AUTOEXEC batch file is *not* the ultimate batch file, but it works. Later, you can come back and improve the file. Once the AUTOEXEC.BAT file works properly, you can go on to configure DOS.

Configuring the Operating System with a CONFIG.SYS File

This section explains how to establish a file that alters some features and functions of DOS. Although this step is not a necessity for starting with a hard disk, you will gain some performance efficiency and convenience from DOS by creating this file. The procedure to customize DOS is both simple and straightforward. The step is as simple as copying a few files and creating one text file.

When you boot your system, DOS invokes the AUTOEXEC.BAT file. (If you turn on your system, the computer first spends some time running POST, the power-up self-test.) However, before DOS works on the AUTOEXEC.BAT file, DOS searches for a special file called CONFIG.SYS, the system configuration file. This file contains a series of statements that tailor DOS to your liking.

A CONFIG.SYS file is not required. Without it, DOS falls back to established default values for certain details. As you gain experience with your computer, you may find that these commands help your system run more efficiently or effectively. Several programs expect more than the DOS minimums and require certain statements in an existing or new CONFIG.SYS file. If you are eager to get started using your hard disk, skip this section and come back later. This section, however, is important to you.

Using the CONFIG.SYS File

CONFIG.SYS is a plain text file. You can create the file using EDLIN, DOS's crude but workable text editor; any word processor capable of producing an ASCII or nondocument file; or the COPY CON technique. Because this file is short, the COPY CON method works well.

The two major commands you should know are BUFFERS and DEVICE. If you are using DOS in a country other than the United States and English-speaking Canada, you should also use the COUNTRY command. The remaining commands are explained but can be skipped now and read later.

Using File Buffers To Increase Disk Performance

The "big" line to put in the CONFIG.SYS file is the BUFFERS statement, which creates buffers in your computer's RAM. This command has the greatest impact on making your disk perform

better. The reason for the increased performance is that information can be transferred much faster within RAM than between the disk and RAM. The speed difference can be 100 to 1.

A disk buffer is a small area of RAM that DOS sets aside. DOS uses the buffer to hold the information to be read from or written to the disk. When the buffer is filled, DOS writes the information to the disk or hands the information to the requesting program.

For example, when a database program calls for information to be read from a disk drive, the information is read from the disk and placed in the RAM buffer. DOS then hands the requested information to the program. As the program calls for more data from the disk, DOS first checks its buffers. If the information is already in this hidden RAM area, DOS does a high-speed memory-to-memory transfer, avoiding the relatively longer time needed to read the information from the disk again.

When a program directs DOS to write information to the disk, DOS hides the information in a disk buffer and then writes the information to the disk. If the program requests this same information and DOS finds the information still in a disk buffer, DOS skips reading the disk and hands the information back to the program. Again, a high-speed memory-to-memory transfer takes place.

If the information isn't in a disk buffer, DOS empties (flushes) the buffer that has not been used in the longest time, goes out to the disk, puts a new chunk of data into the buffer area, and so on. This method of recycling buffers ensures that the most recently used information, which is more likely to be reused, is held in the buffer.

You can see that the more disk buffers DOS has, the more information DOS holds in its buffers and the greater the possibility for the high-speed memory transfers. The technique shines when you are randomly skipping around in a file and reading parts of the file smaller than a disk's sector, the smallest unit for holding disk-based information.

Each disk buffer usually contains 528 bytes, a little more than one disk sector. Unless told otherwise, DOS starts with three disk buffers for a computer with a hard disk, two for machines without. This number is barely adequate. You can specify the number of disk buffers you want by placing the following statement in your CONFIG.SYS file:

BUFFERS = *number*

The value of *number* is the number of buffers you want. The allowable range is 1 to 99.

To a point, the more buffers, the better. The exact number depends on the programs you use routinely. Because disk buffers come from the computer's RAM, each buffer takes away RAM that your programs can use. Since RAM prices have fallen so low, this RAM robbery is less critical, probably not even an issue. Most people have a minimum of 256K of memory, and many have 512K to 640K or more.

If the programs you normally use perform much random disk access (reading data from and writing data to different areas of a file)—many database applications, for example—then increasing the number of buffers will increase performance. Many disk buffers do not perform sequential disk access (reading or writing each section of the file in sequence). DOS's COPY command is an example. COPY never starts in the middle of a file; COPY starts at the beginning and works its way to the file's end. Having many disk buffers does not help programs of this kind.

DOS must do some housekeeping work to manage the buffers, and the work increases as the number of buffers increases. DOS V2 was easily bogged down by too many buffers. DOS V3 handles buffers better, but DOS does reach a point at which the system takes more time to handle buffers than to use the disk drive without buffers. DOS actually slows if you have too many disk buffers.

How many disk buffers should you set up? You should use more than the two or three provided by DOS, but there is no "best" number.

As a guideline, I suggest that you set up between 10 and 40 buffers and experiment. Start with 20 buffers. Because the CONFIG.SYS file is easy to change, try increasing (or decreasing) the number on your system. Remember that DOS does not automatically change itself when you change the CONFIG.SYS file. You must reboot your system to make the changes take effect.

Adding Devices with the DEVICE Command

A device is any peripheral that attaches to your computer, such as a printer, disk drive, video display, and even the keyboard. A device driver is software the operating system uses to control that device.

DOS comes equipped with the software you need to manage the peripherals that come with your computer. To provide for devices that DOS could not anticipate, the DEVICE command was created. For example, DOS does not "know" anything about a mouse (a common pointing device). Without the DEVICE command, you would need to modify DOS to get it to recognize the mouse. With

DEVICE, you simply copy the provided device driver software to your hard disk and add a line to your CONFIG.SYS file. The syntax of the line is

DEVICE = *filename*

The *filename* is the name of the file holding the device driver program. When you boot up your system, DOS invokes the software and alters itself to handle the device. As you add device drivers, DOS grows; the increase in size varies based on the working size of the device drivers.

DOS V2 and later versions come with a prewritten device driver: ANSI.SYS. This driver provides additional capabilities to handle the video display and the keyboard. Controlling the screen's colors and graphics and even the output of each key on the keyboard is easier with ANSI.SYS. To install ANSI.SYS automatically each time you start the computer, you enter in your CONFIG.SYS file a line similar to the following:

DEVICE = C:\BIN\DRIVERS\ANSI.SYS

If you put ANSI.SYS in any location other than the starting directory, you must include a full file specification (the disk drive and subdirectory names) with the driver name. For example, I keep my device drivers in a subdirectory called \BIN\DRIVERS. (See Chapter 5 for a discussion of subdirectories.)

A second device driver included with DOS V3 and later versions is VDISK.SYS, a RAM disk that also can use extended memory on 286/386-based computers (the AT). VDISK is discussed in Chapter 12. The manufacturers provide device drivers for the mouse, expanded memory boards (EMS RAM boards), the Bernoulli Box, some printers, and other devices. With the device drivers, DOS can communicate with, control, and use these devices. You simply copy the file and add to CONFIG.SYS a line for each device.

Internationalizing Your Computer

DOS V2.1 introduced some international features. DOS V3 makes these features easier to set and use.

With the international functions, you can customize various date, time, and punctuation conventions to those used in countries other than the United States and English-speaking Canada. The COUNTRY command is important for anyone using DOS outside the U.S. and

Canada because DOS defaults to United States conventions, which are the following:

Date	mm-dd-yy
Time	hh:mm:ss
Currency symbol	$ to the left of amount
Thousands separator	, (comma)
Decimal separator	. (period)
Data list separator	, (comma)

DOS V3 and later versions can be adapted to the conventions of over a dozen other countries by adding the following line to the CONFIG.SYS file:

COUNTRY = *number*

The value *number* is a three-digit number corresponding to the international telephone code for the country. Table 4.1 lists the codes for countries currently handled by DOS. If you don't add this country-code line, DOS uses 001, the code for the United States.

Table 4.1
Country Codes

Country	*Code*
United States	001
Netherlands	031
Belgium	032
France	033
Spain	034
Italy	039
Switzerland	041
United Kingdom	044
Denmark	045
Sweden	046
Norway	047
Germany	049
Australia	061
Finland	358
Israel	972

DOS also provides the facility to customize the keyboard and display so that you can use the character sets of several foreign languages. This feature is not invoked through CONFIG.SYS but by a separate program. Because the AUTOEXEC.BAT file invokes these

programs at start-up, the commands are discussed in Chapters 7 and 8 on batch files.

COUNTRY is a "free" command. DOS does not increase in size if you use this command.

Using Other Configuration Commands

The total number of CONFIG.SYS commands under DOS V3.2 is nine. The six not yet explained are BREAK, FCBS, FILES, LASTDRIVE, SHELL, and STACK. All six have an impact on how your computer operates. If you are pleased with your setup, you can skip to the end of the chapter and return to these commands later. At some time, however, you should read about the remaining commands. The solutions to several problems lie in using these commands. And some applications programs may explicitly require you to use one or more of the commands.

BREAK. The BREAK command determines when a program or command can be interrupted. You request that DOS halt a program by simultaneously pressing the Ctrl key and the Break key. (The Break key shares a key with Scroll Lock, on the upper right of the numeric keypad.) The BREAK command determines when DOS honors the request.

When BREAK is off (the default), DOS reacts only when the program is working with the screen, keyboard, printer, or an auxiliary device (the device connected to the computer's serial port). When BREAK is on, the BREAK command is honored anytime DOS is called by the program, not just at the times listed.

To start the computer with BREAK turned on, you would add this line to CONFIG.SYS:

BREAK ON

DOS starts with BREAK turned on.

It is usually safe, however, to leave BREAK off (and thus not include a line at all in the CONFIG.SYS file). On the other hand, if you are working with a program that has little interaction with the display, keyboard, printer, or auxiliary device, you might want to turn BREAK on just for the running of that program. To do this, you can invoke the BREAK command from the DOS prompt.

The commands that determine when DOS can interrupt a program are

BREAK ON
BREAK OFF

Then, you can turn BREAK on before running a particular program and turn it off after the program is finished.

BREAK has no effect when the program is computation-bound ("thinking" and not calling DOS). Pressing Ctrl-Break has no effect until the program calls DOS for some activity. Like COUNTRY, this command does not affect the size of DOS in RAM.

FCBS. Programs designed to work with DOS V1 use an outdated DOS mechanism for controlling files. The mechanism is called a File Control Block (FCB). DOS V2 handles FCBs without restriction. DOS V3 allows FCBs but restricts the number of FCBs that can be used at one time.

If you see the message

```
FCB unavailable
Abort, Retry, Ignore
```

you have no more File Control Blocks. This message appears only with programs that use the old style of FCBs and file-sharing (networking) software. Without file sharing loaded, the program exceeding the FCB limit malfunctions and usually displays some error message about not finding a needed file.

To increase the number of FCBs in DOS V3 and later versions, use the following command:

FCBS = maxopen, *neverclose*

The value of *maxopen* is the maximum number of unique File Control Blocks DOS allows a program to use at one time. Some programs have a bad habit of not freeing unneeded FCBs. For this reason, DOS can automatically recycle FCBs not used for the longest time. The value in *neverclose* restricts the number of FCBs that DOS can automatically reuse.

The default value for both *maxopen* and *neverclose* is 4. You can specify any number from 1 to 255 for *maxopen*; *neverclose*, if specified, must be equal to or less than *maxopen*.

Although the use of FCB is fading fast, some older programs still employ this method of accessing files. (One clue is that a program knows nothing about subdirectories.) If programs that worked under DOS V2 suddenly die under DOS V3, use the FCBS command. FCBS = 12,12 frequently works, but the programs you used may require a higher number. The best method is to experiment.

FCBS does change the size of DOS. Each FCB specified by a *maxopen* over 4 increases DOS by about 40 bytes.

FILES. FCBS is the old way for programs to work with files. The newer way is patterned after the methods used in another operating system, UNIX. This method uses a *handle*, a two-byte number DOS assigns to the file or device the program uses. Each device or file opened by the program has one handle.

The FILES command in the CONFIG.SYS file specifies the maximum number of file handles that can be used at one time. The syntax of the command is

FILES = *number*

The value of *number* is a number from 8 to 255 for DOS V3, 5 to 255 for DOS V2. If you don't include this statement (or if you specify a number less than 8 or 5, respectively), DOS defaults to 8 or 5 (depending on the version of DOS). Each file handle over 8 (or 5) increases DOS's size by 48 bytes.

Many programs request that you use more than the default number of file handles. For example, Brief™ (a text editor), dBASE III Plus®, and NewWord® (a word processor) request 10 to 20 file handles. Omitting this command causes the programs to "squawk" (give an error message) and quit. Your program's documentation will specify the correct value for FILES. If the default value of 8 does not work, a value of 20 should.

LASTDRIVE. With the LASTDRIVE command (appearing for the first time in DOS V3.0), you can specify the maximum number of disk drives on your system. The syntax of the command is

LASTDRIVE = *x*

The letter *x* is the alphabetical character for the last possible disk drive on your system. Any letter up to Z can be used. If you do not use this command in the CONFIG.SYS file, DOS "assumes" that the last disk drive is drive E.

At first, this command sounds less than useful. If you are using the common configuration of one or two floppy disk drives and a hard disk (three disk drives), how do you use the nonexistent drives D and E? The useful occasions for LASTDRIVE occur when (1) you add to your system certain devices (like a tape backup unit) that "look like" a disk drive to DOS; (2) you need a "phony" disk drive letter for the DOS SUBST command, which makes a subdirectory "look like" a separate disk drive; or (3) your PC is tied to a microcomputer network.

The SUBST command is described in Chapters 7, 8, and 14. On other occasions, consult your equipment's documentation for the

proper value for LASTDRIVE. This is a "free" command; LASTDRIVE does not increase the size of DOS.

SHELL. The SHELL command is meant for more advanced DOS users. It is probably best for you to use it later, if needed. SHELL works slightly differently for each version of DOS (V2.0, V2.1, V3.0, V3.1, and V3.2).

Originally, the SHELL command was intended for those wanting to use another command processor instead of the COMMAND.COM file. COMMAND.COM is the part of DOS that displays the system prompt (A>, C>), contains some built-in commands, and runs your programs. Microsoft, the authors of DOS, assumed that some programmers might want to build and use their own command processors.

The phrasing of the basic SHELL command is

SHELL = *d:path\filename.ext*

The *d:* is the optional disk drive name, *path* is the optional directory path, and *filename.ext* is the name of the new command processor. (The file extension is optional unless it is other than COM.) The basic SHELL command works on all versions of DOS.

(Note that the syntax of the SHELL command includes a reference to a *path* in which the command processor could be located. The subjects of paths and subdirectories are important enough to warrant separate and extensive coverage in Chapters 5, 6, and 7. For the moment, suffice it to say that a *subdirectory* is a hierarchical scheme for subdividing a hard disk, and a file's path is the chain of subdirectories leading to that file.)

The other purpose of the SHELL command is to let you place the command processor, including COMMAND.COM, in a different directory. This feature works on IBM PC DOS V2.1 and later, but not on some versions of MS-DOS V2.1. The full SHELL command works differently on the different versions of DOS.

Before explaining the other syntaxes for SHELL, I should mention that all information put in the CONFIG.SYS file with SHELL is important, including spacing and punctuation. DOS is very fussy about the phrasing of the SHELL command. Inserting a space where none belongs, forgetting a space where at least one must be given, or forgetting or misstating a name causes DOS to lock up. If you mess up the command, reboot your system using a DOS diskette. You'll need to correct your CONFIG.SYS file before you can restart DOS from the hard disk.

The full syntax of SHELL for DOS V2.1 and V3.0 is

SHELL = *d:path\filename.ext d:path\ /P*

The first part of the command is the same as given previously. The addition is the second *d:path*\, which must be identical to the first *d:path*\. The second drive and path are used by DOS to remember where the command processor is kept. If this information is inaccurate, DOS gets lost.

The */P* means "stay permanent." If this part is not given, COMMAND.COM starts up, executes the AUTOEXEC.BAT file, then exits. The result is a computer "left dead in the water."

You can use or omit spaces around the equal sign or add extra spaces between items. Don't add spaces within the drive, path, and file names and don't forget to use at least one space between the first and second names and before the */P*. The rule about upper- and lowercase still applies. DOS does not care.

DOS V3.1 and V3.2 add one more option to SHELL. DOS maintains something called the *environment*, a part of RAM where information can be stored and later read by your program. You place information in the environment by using the SET command, covered in Chapter 8. You don't have to understand what the environment is for now, only that if you see the following message, you need to increase the environment's size.

```
Out of environment space
```

The SHELL command for DOS V3.1 and V3.2 is

SHELL = *d:path\filename.ext d:path\ /P /E:size*

The addition is */E:size*. The */E:* stands for environment. The value of *size* is the amount of space set aside for the environment. The number for *size* differs between DOS V3.1 and later versions. Increasing the size for the environment also increases DOS's size.

For DOS V3.1, *size* is the number of 16-byte sections (sometimes called paragraphs) of RAM reserved for the environment. DOS V3.1 and earlier start with a 128-byte environment. The allowable range for *size* is between 11 and 62, meaning an environment size of 176 bytes to 992 bytes. Using a *size* of 20 gives an environment space of 320 bytes (16 * 20). DOS displays an error message and ignores */E* if you use an incorrect value.

For DOS V3.2 and later, *size* is the number of bytes for the environment. To specify a 320-byte environment, you use */E:320*. The allowable range for *size* is 160 (the new default environment size) to 32,768. The number should be evenly divisible by 16; if not, DOS rounds to the next higher multiple of 16. If you specify a number greater than 32,768, DOS uses a *size* of 32,768.

The order of */P* and */E* can be reversed, and the space between the two can be omitted. You can omit the */E* parameter, if desired. However, don't forget the colon between the */E* and *size* and don't put a space within any part of the */E:size.*

Who should use the SHELL command? Unless you are exceptionally gifted with assembly language programming, you should forget about writing your own command processor; but you may have other occasions to use SHELL.

If you want to move COMMAND.COM to a directory other than the starting directory, you must use the SHELL command. However, some DOS programs in V3.1 and earlier, such as FORMAT, demand COMMAND.COM in the starting directory. You don't get a choice.

If you need to increase the size of the environment, you must use the SHELL command. Remember that only DOS V3.1 and later allow this option. Also remember that DOS V3.1's SHELL command is slightly different from later versions. You probably won't run out of environment space for a while, so you can skip this command and use it later when the need arises.

If you don't fit in either category but you insist on experimenting, try the command on a *copy* of your DOS System disk. Remember that specifying a bogus SHELL command will "freeze" your system.

STACK. The STACK command is an addition to DOS V3.2. If you read your DOS manual for the SHELL command, you'll note that it is not fully discussed. Options are missing. STACK is a new CONFIG.SYS command and is mentioned only in the appendix of the DOS manual. Explaining STACK requires a brief technical look at the computer.

A *stack* is an area of RAM. The CPU (central processing unit, the "heart" of the computer) uses this area for temporarily holding information. High-speed instructions push items on to the stack and pop items from the stack. Most eight-bit computer systems use one fixed-location stack. The CPU of the IBM and compatibles can use a stack almost anywhere in memory, and each program can have its own stack.

A *hardware interrupt* is a signal generated by a device demanding attention. For example, each time you press a key, the keyboard generates an interrupt. The same is true for disk drives, the communications ports, a mouse, or any other device. The CPU executes software in a fixed location to satisfy the interrupting device. Interrupts can *nest*, meaning that the computer can be interrupted while handling a different hardware interrupt.

A stack is established for each program you run. DOS maintains its own pool of stacks, using one for itself and reusing the remaining stacks when interrupts must be handled.

PC DOS V3.1 and earlier had no limit to the number of simultaneous stacks. DOS V3.2 has a limit of nine. DOS reserves one stack for itself, so eight can be used for handling interrupts. If the ninth interrupt occurs before DOS can finish handling any of the previous eight, DOS displays the following message:

```
FATAL: Internal Stack Failure, System Halted
```

The system stops. The computer must be turned off and on again to restart DOS. Ctrl-Alt-Del, the system reset sequence, doesn't work.

This problem isn't difficult to reproduce. Holding down a key while some programs are working triggers the stack failure. The problem affects the newer Selectric-like keyboards more than the older keyboards.

To solve (or prevent) this problem, use the STACK command. The syntax is

STACKS = *number, size*

The value of *number* is the number of stacks, and *size* is the size of each stack. The default value of *number* is 9 and can range from 8 to 64. The value of *size* defaults to 128 bytes and ranges from 32 to 512. Increasing or decreasing the number of stacks and their size increases or decreases the size of DOS.

The recommended procedure is to increase the number of stacks and not alter their size. If you must alter the size, increase it; don't decrease it. A comfortable number is *12, 128*: 12 stacks of 128 bytes each.

Creating Sample CONFIG.SYS Files

This section presents three model CONFIG.SYS files. The first example is a minimal CONFIG.SYS. If you don't like the other two examples, use the first example.

You will need a text editor or a word processor capable of creating an ASCII or nondocument text file, or you may use the command

COPY CON: C:\CONFIG.SYS

This line places the information into the configuration file.

The first example simply establishes a comfortable number of disk buffers and file handles. The text is

BUFFERS = 30
FILES = 20

Type these lines. If you are copying from the keyboard to CONFIG.SYS, don't forget to press the F6 and Enter keys to inform the COPY command that you are finished.

The second example adds the ANSI.SYS file. Before using this configuration file, you need to perform three additional steps. Type

MKDIR C:\BIN
MKDIR C:\BIN\DRIVERS

These commands will be explained in Chapter 5.

Make sure your DOS Master diskette is in drive A and the door is shut. Type

COPY A:*.SYS C:\BIN\DRIVERS

This command copies all .SYS files (all device driver files) to the subdirectory you'll use for device drivers.

The finished model CONFIG.SYS file is

BUFFERS = 30
FILES = 20
DEVICE = C:\BIN\DRIVERS\ANSI.SYS

The third example is an all-options CONFIG.SYS. This file employs every CONFIG.SYS command except BREAK. The full CONFIG.SYS file is

BUFFERS = 30
FILES = 20
FCBS = 22, 22
LASTDRIVE = I
STACKS = 12, 256
DEVICE = C:\BIN\DRIVERS\ANSI.SYS
SHELL = C:\COMMAND.COM C:\ /P/E:480

This file establishes 30 disk buffers, 20 file handles, and 22 FCBs that cannot be closed. The file states that the last logical disk drive is I, establishes 12 DOS stacks of 256 bytes each, and loads the ANSI.SYS device driver from the directory of C:\BIN\DRIVERS. Finally, the file loads COMMAND.COM from the starting directory of drive C, setting up an environment size of 480 bytes.

An environment size of 480 bytes is for DOS V3.2. DOS V3.1 should use /E:30 as the last part of the SHELL command instead of /E:480.

The third example is recommended for those using 512K or more of memory. All the options, including the use of ANSI.SYS, take up 24K more of RAM. If you are very tight on memory in your 512K or more system, reduce or omit the FCBS command and change STACKS to 12, 128.

To test the file, boot DOS from the hard disk. Watch for any message that states something like "command ignored" or "bad command." If you get such a message, check your CONFIG.SYS file. You may need to boot DOS from a floppy diskette to restart the system. Make the needed changes and try again.

Summary

This chapter covers the physical (get it plugged in) and logical (get it set up with DOS) installation of the hard disk. Although the procedure you use may vary slightly, this chapter covers most of the ground. In this chapter you learned the following major points:

- Preparing a hard disk requires three major steps: physical formatting, partitioning, and logical formatting. FDISK partitions the disk. FORMAT logically formats the disk.
- The AUTOEXEC.BAT is a batch file DOS automatically uses when DOS is started or restarted.
- CONFIG.SYS is the system configuration file, which you use to tailor DOS for your needs.

This chapter ends the background, selection, and setup sections for hard disks. Your hard disk should be ready to use. However, more work is needed to make your hard disk, DOS, and your programs more efficient, easy, and intuitive to use. This process is the subject of Part 2.

Part II

Hard Disk Management

Part I concludes with the discussion of installing your hard disk; it is ready for use.

The ease and efficiency with which you use your hard disk depend on your ability to exploit an important DOS feature: the hierarchical directory system. Hierarchical directories are used to group and organize files. If you understand and use hierarchical directories, hard disk management is easy and disk operations are effective.

If you use the system ineffectively, your computer use is less effective and more frustrating. If you do not understand the system at all, your computer use is ineffective and inefficient; the computer manages you instead of your managing the computer. I cannot understate your need to understand how hierarchical directories operate. You must know what directories are, how they can be tiered, and how the path is used. If you do not understand these concepts, you cannot effectively use your hard disk.

Chapter 5 presents an overview of the essentials of the directory system. The chapter gives you the information you need in order to command and control the system. Chapter 6 reviews several familiar DOS commands and relates how these commands are used with hierarchical directories.

Chapters 7 and 8 present information on batch files, the method used to automate DOS tasks. Chapter 7 discusses the basics, and Chapter 8 covers the commands unique to batch files. In Chapter 8 also are many examples that can be adapted to your particular programs and needs.

The material in Chapter 9 summarizes and extends the information presented in earlier chapters of this part. Chapter 9 provides general guidelines for organizing programs and files, starting with the DOS files and continuing through several major applications programs. Some additional DOS commands and techniques are introduced for working with hierarchical-hostile programs; that is, programs that are difficult to use with the hierarchical directory system.

Newcomers should read both Chapters 5 and 6. The two chapters are the "Rosetta Stone" for the hierarchical directory system. If you do not understand the information in Chapter 5, move on to Chapter 6 and examine the examples, which show some practical applications of the commands. You can learn the system by experimenting with the examples. Once you understand the basics of hierarchical directories, you can master your hard disk.

Reading Chapter 7 will help new users understand batch files and the way they work. Chapter 8 expands your knowledge of batch files and presents both common and sophisticated techniques. Understanding how to invoke programs through a batch file is the most important concept. The remaining information can be reviewed at your leisure.

Experienced users can skim Chapters 5, 6, and 7. Consider these chapters as refreshers on hierarchical commands and batch files. In Chapter 8, however, experienced users will find some interesting techniques on advanced batch files. Chapter 9 is important for all readers because it sums up the concepts presented in this part and presents the hard disk "philosophy" used throughout this book.

Mastering the concept and techniques of hierarchical directories, as I have stressed, is an essential key to effective and efficient use of your hard disk.

Chapter 5

Hard Disk Organization and Hierarchical Directories

The preceding chapter outlines the minimum steps for setting up your hard disk. To use the hard disk effectively, you must perform some additional preliminary steps. The work, however, is painless and rewarding. Soon working on the hard disk will become intuitive, second nature; your daily tasks will be completed faster; and you will avoid certain pitfalls.

This chapter introduces you to the mind-set for using a hard disk. In this mind-set, you organize by purpose, keep handy tools nearby (useful programs on the hard disk), and employ imaginatively the facilities DOS provides. The secret, if this mind-set can be attributed to any single element, is to use subdirectories. But first it's important to take a look at the rationale for abandoning the floppy mind-set.

Abandoning the Floppy Mind-Set

The primary advantage of a hard disk is its storage capacity. A great many files can be stored on a single hard disk. With increased capacity, however, also comes the potential for utter chaos.

As discussed in Chapter 2, DOS places limits on the size of the starting directory of the diskette. Floppy diskettes can hold no more

than 112 or 224 files in the starting directory. This maximum is a physical limitation of DOS. You gain no advantage by circumventing the limitation. If you increase the number of files, DOS cannot perform efficiently.

Diskettes also have a fixed amount of storage. Your floppy diskettes can hold comfortably 60 or 70 files. The list of files (called up by the DIR command) may be awkward, but the list is still manageable. DOS quickly locates and uses the correct file.

What happens if you add unlimited files to that floppy diskette? When you add the 89th file, are files as easy to locate? Add the 489th file—is the task of locating a file any easier? Of course not. After all, you are committing an organizational sin—the electronic equivalent of storing *all* your paper files in one huge container labeled "File Drawer." The ease of filing is unmatched; the task of retrieval would cause the resignation of the most patient file clerk.

In the same way, adding files to a diskette makes keeping track of the files more difficult and slows retrievals. Just as a file clerk must search through more files, so must DOS. Overall disk operations become slower.

When you store thousands, not hundreds, of files, you must organize differently. Subdividing your file drawer by subjects or with A-through-Z headings makes sense. You locate the subdivision and then locate the file. You do not search through hundreds of individual papers, just through a set of dividers. When you find the right divider, you search for the specific file.

You use this same approach with floppy diskettes. You place on a single diskette the files related to a certain task or program. For example, one diskette may hold all your working DOS programs. Another diskette holds a spreadsheet or a word-processing program. Still another set of diskettes holds your spreadsheets or documents. You also can categorize the spreadsheets or documents into groups: by date, by client, by purpose, and so on.

People often lose this sense of organization when they move files to a hard disk. They treat the hard disk as if it were one big floppy diskette. Users attempt to dump everything into one directory, forgetting that a hard disk can be organized like a set of floppy diskettes, with divisions based on purpose. The approach that must be employed is making and using multiple directories: you must use subdirectories.

Organizing the Hard Disk into Subdirectories

The secret to using the hard disk effectively is creating subdirectories. All versions of DOS, starting with V2.0, offer the organizational tool of *hierarchical directories*, a method for grouping files by application or other characteristics. Each group of files is placed in its own directory.

As implied by the word *hierarchical*, this system of directories is analogous to a tree structure or, more accurately, a root structure. Imagine the root structure of a tree growing out from the main tree trunk (see fig. 5.1). Each branch lower in the hierarchy grows out of one and only one higher branch. However, any branch can have more than one lower branch growing from it.

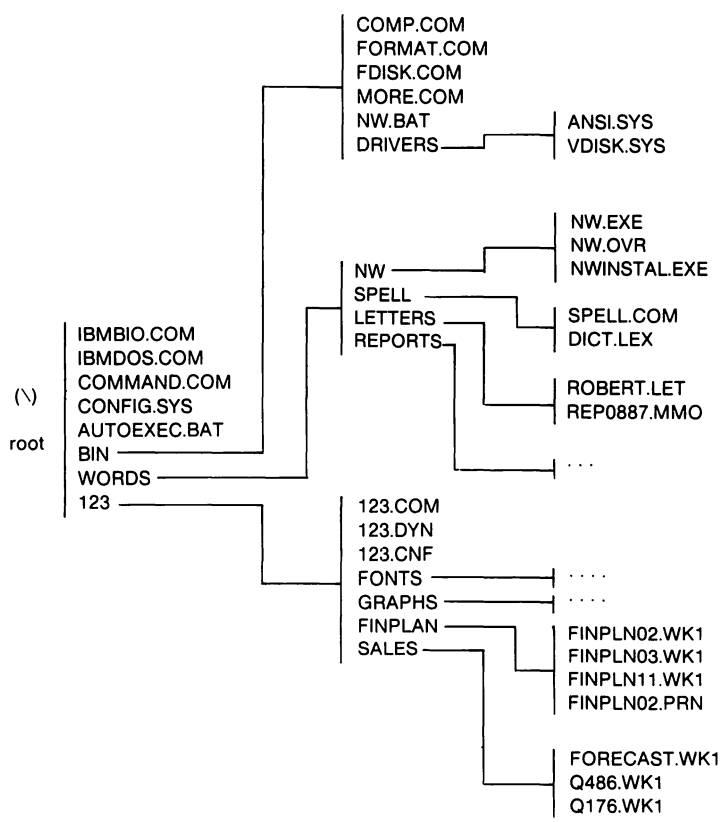


Fig. 5.1. *Tree structure of a hierarchical directory.*

Another analogy can be made to a family tree. A lower branch is referred to as the "child" of a higher branch; the higher branch is called the "parent" of a lower branch. Every child directory has only one parent. However, a parent directory may have several children.

The eldest (highest) level in the hierarchy is called the *root directory*. The root directory is designated by a single backslash (\). This directory is the starting directory of the disk—the only directory created when you format a disk. All child subdirectories trace their heritage to this top directory. In figure 5.2, XYZ is the parent of ABC and DEF; ABC and DEF are the children of XYZ.

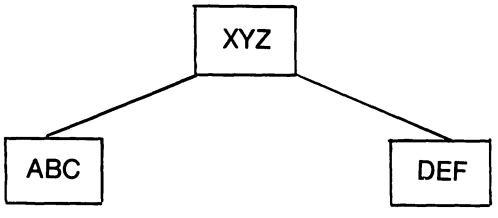


Fig. 5.2. *Child subdirectories.*

The key to the hierarchical directory system is the fact that directories can hold more than files; directories can hold other directories. This concept is the same as having file folders within file folders. The file folder can hold papers or other file folders. In turn, each inner file folder can hold more papers or more file folders. This basic structure allows a multilevel organization, and the organization of the filing system is based on whatever criteria you desire. A typical example might be the tree structure shown in figure 5.3.

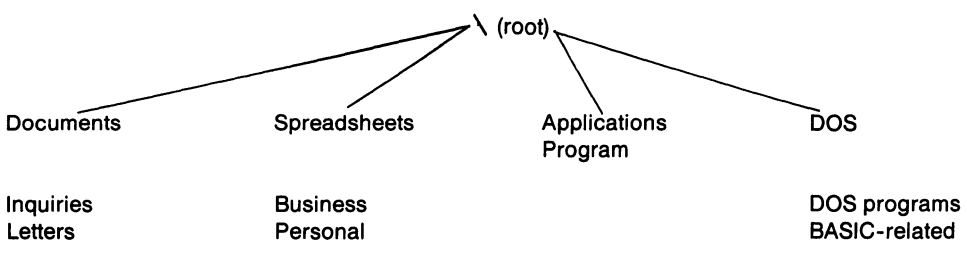


Fig. 5.3. *Function-oriented tree structure of hierarchical directories.*

The structure in figure 5.3 is function oriented. The files are organized by the major programs used and subdivided within the major directories by data file type: letters, business spreadsheet models, personal spreadsheet models, DOS programs, and so on. Another approach is to divide the disk by "real world" applications and then divide by program function, as in figure 5.4.

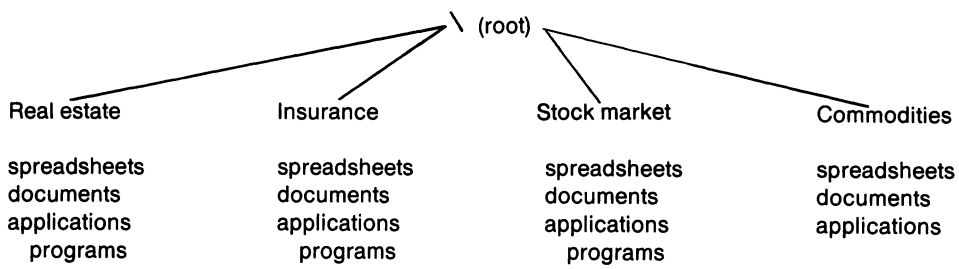


Fig. 5.4. *Application-oriented tree structure.*

No one way of subdividing is best. The most common organization is a hybrid: some division by program and some by application. Suggestions for organization are given in Chapter 9 and Chapter 17 on setting up major programs.

The number of subdirectories you should create does have practical limits. Because DOS and you "see" only a subset of files at one time (the files in the subdirectory), too many subdirectories create confusion and headaches. You must keep track of both the files and their respective subdirectories. Instead of shuffling floppies, you shuffle subdirectories. However, tools (programs) are available to help you with the directory shuffle. These programs are discussed in Chapter 11.

DOS has one physical limit to the number of subdirectories. That limit is disk space. Although subdirectories take little disk space (usually one disk cluster), subdirectories do vie with your files for free space. The amount of storage and the use of disk space by other files determine the physical limitation.

Before learning to use subdirectories, you should know another conventional term: *level.* This word refers to the number of layers from the root directory to the subdirectory where a file is located.

The root directory, which is not contained in any other directory, can be considered at level zero. Usually, the root directory is not viewed as a level but simply as the root. Any subdirectory located in the root directory is considered a first-level directory. In the example in figure 5.4, Real estate, Insurance, Stock market, and Commodities are level-one directories. A subdirectory of any of these four directories is a level-two subdirectory. Each new layer of subdirectories is a new level.

In this book a subdirectory is often called a directory. The two terms are interchangeable, except for the root directory. Every directory except the root is owned by another directory and also holds the names of files and other directories. However, the root directory has no parent and is therefore not a subdirectory. Except for this difference, the terms *subdirectory* and *directory* are identical.

Subdirectories also can be used on floppy diskettes. Because subdirectories take some disk space and floppy diskettes do not hold all that many files, subdirectories are less frequently used on floppy diskettes. However, subdirectories are used to store many small files that exceed the number of files allowed in the root directory. Subdirectories also are used on diskettes to organize files into sets, particularly when the files will be transferred by set to the hard disk.

Understanding What DOS "Sees"

Hierarchical directories help organize information. This statement is based on how DOS "sees" each directory. When you are "in" a particular directory, DOS "sees" only the files contained in that directory. This single attribute offers great advantages in organization.

Suppose you have in the root two subdirectories: SHEETS and LETTERS. When you do a directory of your hard disk, you see the two subdirectories listed as follows:

```
Volume in drive C is DOS
Directory of C:\

COMMAND  COM
     .
     .
     .
SHEETS   <DIR>
LETTERS  <DIR>
     .
     .
     .
```

DOS is myopic. From the root directory, DOS sees only the files and subdirectories within the root directory. DOS does not see the files (or other subdirectories) in the subdirectories the root holds. Unless you explicitly lead DOS down the path to those subdirectories of subdirectories, DOS doesn't see the other files. For DOS to see the list of files in the SHEETS subdirectory, you type

DIR C:\SHEETS

This command directs DOS explicitly to display the list of files from the subdirectory SHEETS, which is named in the root directory. The listing may be something like the following:

```
Directory of C:\SHEETS

.          <DIR>
..         <DIR>
more files here
```

Notice the first two entries in the directory listing. Every subdirectory has these two dot entries. The root, which is not a subdirectory, does not have these entries. The section "Changing the Current Directory" explains what the dots mean and how they can be used.

You also need to maneuver DOS through the directories to run a program that is not in the directory you are currently using. For example, if your DOS files are in a separate directory called BIN, you invoke the check disk (CHKDSK) program by typing

C:\BIN\CHKDSK

After CHKDSK has run, DOS returns to the directory you were using before you ran the command. This method of invoking programs is

available in DOS V3 and later. DOS V2 users do not have this handy feature (an encouragement to upgrade from DOS V2).

Understanding the Path

Chapter 2 covers names of files, disk drives, and devices. When specifying a file, you may use a root name followed by an optional extension and preceded by an optional disk drive name. This definition was simple but incomplete. The preceding section quietly introduced the omitted part: the path.

The DOS term *path* has three definitions. Each definition involves directories. A *path* may be defined as

1. The names of the chain of directories leading to a file
2. The complete file or directory name
3. A DOS command

Understanding paths is essential to effective use of subdirectories. This book uses the first definition of path unless otherwise specified.

Figure 5.1 shows a sample hard disk hierarchy for drive C. At the top is the root directory. The root owns (holds) several files, such as IBMBIO.COM, CONFIG.SYS, and AUTOEXEC.BAT. The root also owns several subdirectories: BIN, WORDS, 123 (spreadsheets), and others.

Each subdirectory in the root directory holds additional files and subdirectories. For example, BIN holds common DOS programs and batch files and the DRIVERS subdirectory. In turn, the DRIVERS subdirectory holds some device driver files, such as ANSI.SYS for the console and VDISK.SYS for a RAM disk (see Chapter 12 for a discussion of VDISK).

The WORDS directory holds four other subdirectories: NW (word processor), SPELL (spelling program), LETTERS, and REPORTS. Each subdirectory holds several files. 123, the spreadsheet directory, holds several subdirectories (FINPLAN, SALES, and others). The 123 subdirectory also has several program files in the directory and uses additional subdirectories for each major group of program files (FONTS and GRAPHS).

Elements of the Path Name

Suppose you wish to access the file FINPLN02.WK1. To maneuver to FINPLN02.WK1 from the root directory, you move from the root

to 123 to FINPLAN, and in FINPLAN you access FINPLN02.WK1. The chain of directories to the spreadsheet is

root ⟶ 123 ⟶ FINPLAN ⟶ FINPLN02.WK1

To tell DOS how to execute this move through the directories, you specify a path. The path for this example is diagrammed as

directory	root ⟶	123 ⟶	FINPLAN	⟶ FINPLN02.WK1
elements	\	+ 123	+ FINPLAN	⟶ FINPLN02.WK1

The rules for building a path name are as follows:

1. The path name traces the movement from the root directory to the directory holding the file.
2. The directory names are joined by a backslash between each pair of names, except after the root directory. Because the character representing the root directory is itself a backslash, another backslash is unnecessary.
3. When you use a file name with the path name, join them by placing a backslash after the last directory name in the path and adding the file name, unless this last directory is the root directory.

Putting these rules together, the proper phrasing of the path name for FINPLN02.WK1 is shown in figure 5.5. The process is outlined as

directory	root ⟶	123 ⟶	FINPLAN	⟶ FINPLN02.WK1
elements	\	123	\ FINPLAN	\ FINPLN02.WK1
path name	\123\FINPLAN			

The final form for the path and file name is

\123\FINPLAN\FINPLN02.WK1

On the other hand, the proper path for COMMAND.COM is simply

C:\COMMAND.COM

Remember that you do not use an additional backslash when separating the root directory name from any other directory or file name. The phrasing is *not* C:\\COMMAND.COM.

Many newcomers stumble over the dual use of the backslash, commonly called the *path character*. The location of the backslash in a path name determines the character's meaning. When a path name *starts* with the backslash, the starting point for your movement

through the directories is the root directory. If the backslash is used anywhere else in the path, the backslash is simply a name separator. When you need to begin at the top (the root), start the path with a backslash. Any time you need to separate directory names within a path or separate a path name from a directory name, also use a backslash.

DOS imposes a logical limit on the length of a path name: 63 characters. Included in this 63-character limit is any given file name and the backslashes. The longest file name is 12 characters (8-character root name, 3-character extension, plus the period). When you subtract the length of the file name, the maximum path name length is 50 characters. Because most people tend toward shorter directory names, the 63-character limit is seldom reached.

The Complete File Name

At this point, you have the information needed to describe a complete file name. A complete file name consists of a disk drive name, a path name, and the file name. The general form is

d:path\rootname.ext

The *d:* is the disk drive name, *path* is the directory path (path name) to the file, and *rootname.ext* is the name of the file.

This form is frequently used when specifying a file name to DOS. Accordingly, the terms *file specification* or *filespec* are used in the DOS documentation. The terms *complete* or *full file name* and *file specification* have the same meaning. For example, the complete file name for FINPLN02.WK1 is diagrammed in figure 5.5.

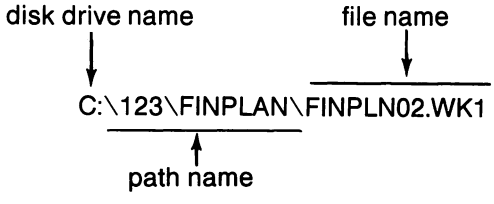


Fig. 5.5. *A complete file name.*

The complete file specification for ROBERT.LET is

C:\WORDS\LETTERS\ROBERT.LET

Using figure 5.1 as a guide, examine how the following file specifications are built:

C:\AUTOEXEC.BAT
C:\BIN\FORMAT.COM
C:\BIN\DRIVERS\ANSI.SYS
C:\WORDS\NW\NW.EXE
C:\123\123.CNF
C:\123\FONTS\BOLD.FNT

Figure 5.1 has a total of 14 directories: the root directory and 13 subdirectories. Often more subdirectories are used on a hard disk. Too many subdirectories can cause problems, but many more subdirectories could be added to this example.

Creating, Changing, and Deleting Subdirectories

This section covers the commands that create, delete, and change subdirectories. The information also extends the concepts of current disk drive to the hierarchical directory system.

Creating a Subdirectory

The first subdirectory command is MKDIR, or in its shorthand form, MD. This command "makes" a subdirectory. The phrasing is simple:

MKDIR *d:path***directory_name**

or

MD *d:path***directory_name**

Both forms are acceptable and equal to DOS. The *d:* is the name of the disk drive where the new directory will be created. The *path*\ is the chain to the subdirectory in which the new directory will be made. The **directory_name** is the name of the directory that will be created.

This command form follows the conventions established in the Conventions section and used throughout this book. The words printed in boldface (the command name, MKDIR or MD, and the

directory name) must be given. The parts of the command printed in italic type (the disk drive and path name in the preceding example) are optional. When you omit an optional designation, DOS uses the current one.

A directory name follows the same rules as file names. Alphanumeric characters and certain punctuation characters are accepted. Other characters, including the wild-card characters, and spaces are not allowed. A directory has a root name and an optional extension separated by a period. The root name has a limit of eight characters; the extension limit is three characters. Because of the 63-character limit on path names (and people seldom enjoy typing long path names), extensions are seldom used in directory names.

The following commands created the three level-one subdirectories shown in figure 5.1.

MKDIR \BIN
MD \WORDS
MD \123

Notice that the path always starts with a backslash, indicating that the subdirectories are to be created in the root directory. No backslash is used to separate the root directory name from the new subdirectory names.

To make the second-level subdirectories of WORDS, you would enter these commands:

MD \WORDS\NW
MD \WORDS\SPELL
MD \WORDS\LETTERS
MD \WORDS\REPORTS

In this case the path is \WORDS, and a path character is used to separate the path from the new subdirectories. When you add directory names other than the root, separate each pair of entries with a backslash.

Because DOS treats subdirectories and files similarly, you cannot create a subdirectory that has a name identical to an existing file in the same directory, nor can you create a file that has a name identical to an established subdirectory in the same directory. Unless some part of the two names is different, DOS refuses to honor the second name.

For example, in the root directory, you cannot create a directory named AUTOEXEC.BAT. Because you seldom use extensions in directory names, this restriction is usually not a problem. A good

practice, however, is never to use a directory name that is identical to the root name of a file in the same directory. For example, don't create a level-one subdirectory called AUTOEXEC. This name could lead to confusion.

An acceptable and common practice is to put a file with the same root name *in* a subdirectory with the same name. For example, C:\WORDS\NW holds a file with the name NW.EXE. C:\123 has 123.COM. Because the file and the directory are at different levels in the hierarchy, no conflict occurs.

Changing the Current Directory

Chapter 2 explains that if you don't give DOS a specific disk drive, the system uses the current disk drive. DOS also has current directories. Like the current disk drive, the *current* (or default) *directory* is the directory that DOS uses unless you specify otherwise. If you omit the path name, DOS uses the current directory.

The current directory is slightly different from the current disk drive. Although your system has only one current disk drive, each disk drive has its own current directory. Changing the current directory on drive C does not affect the current directory on drive A or any other drive.

When you start your system or change floppy diskettes, the current directory is the root directory of the disk. To change directories, you use the *change directory* command. The syntax is

CHDIR *d:***path**

or

CD *d:***path**

When you issue either form of the command, the last directory named in *path* becomes the current directory. If you omit the disk drive name, the current directory on the current disk drive is changed. If you include the disk drive name, the current directory on that disk is changed.

With the CHDIR command, you can jump into any subdirectory and use it as your "base of operation." For example, to move from the root directory to 123, you type

CHDIR \123

or

CD \123

Now, if you issue the DIR command, DOS displays a listing of the files in the current subdirectory, 123 (see fig. 5.6).

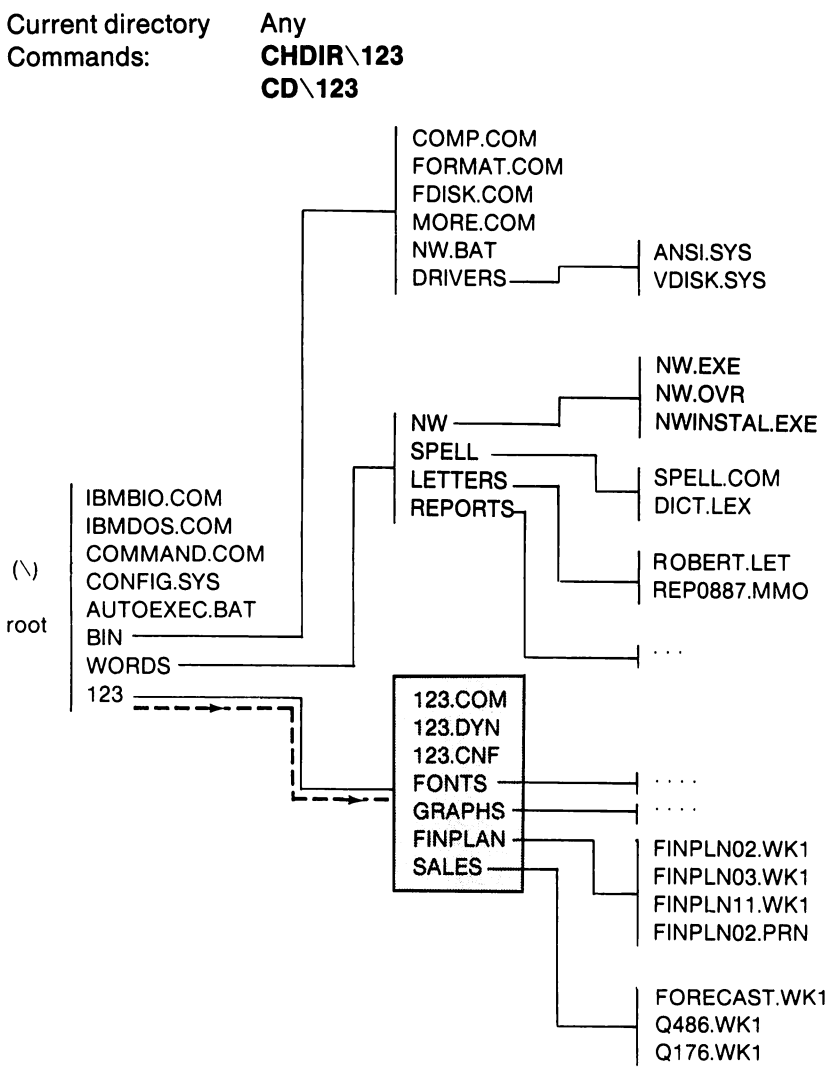


Fig. 5.6.

As with the MKDIR command, if you start the path with a backslash, DOS interprets that to mean that you wish to start the move from the root directory. Otherwise, DOS starts the move from the current directory.

Using Absolute and Relative Paths

DOS has two types of paths: absolute paths and relative paths. *Absolute paths* start at the root directory. The first character is always the backslash. *Relative paths* begin in the current directory. The path to a particular directory depends on the current directory.

Two special characters are used frequently in relative paths. These characters are the period (.) and double period (..), usually called the *dot* and *double-dot*. To see these characters, enter the following commands:

**CD **	Ensure that you are in the root directory
MD TMP	Make a temporary directory called TMP
CD TMP	Move into the temporary directory
DIR	Get the list of files in TMP

You should see a directory that looks something like this:

```
Volume in drive C is DON'S DISK
Directory of  C:\TMP

 .          <DIR>      8-11-86  12:00p
 ..         <DIR>      8-11-86  12:00p
        2 File(s)    921600 bytes free
```

The dot (.) is the "I exist to myself" entry. In a command, the dot is the shorthand form of designating the current directory. The double-dot is the shorthand form of designating the parent directory. Essentially, the double-dot tells DOS to jump up one directory level and go to the directory that owns the subdirectory you are operating in.

Each subdirectory has these two entries. When DOS creates a subdirectory, DOS uses some disk storage to keep track of the hierarchical sequence of directories. However the dot and double-dot are not true files. You cannot erase or rename either entry.

In the commands you just issued, the first command absolutely moved you to the root directory, even if you were in that directory when you started. The second and third commands (MD and CD) use relative paths. The MD command made a subdirectory called TMP in your current directory (the root). The CD command made TMP the current directory. When you asked for a directory, DOS displayed the directory of TMP.

To move back to the root directory, you have two options. You can use an absolute or a relative path. The absolute path command is

**CD **

The relative path command is

CD ..

In this case, the absolute path is one character fewer to type. If your current directory is SALES and you want to make 123 your current directory, you type one of the following:

CD \123

CD ..

The second form is easier to use than the first (see fig. 5.7).

To move from LETTERS to REPORT, you can enter either of the following:

CD \WORDS\REPORT

CD ..\REPORT

The latter is easier to type. Generally, when the current directory is several levels from the root, backtracking a level or two in the hierarchy using the double-dot command is easier than giving a path name starting at the root (see fig. 5.8).

You may have noted that you need not include the backslash as the first character in the path when the current directory is the root directory. You do not need to tell DOS to back up to the root directory when you are operating from the root.

Finding the Current Directory

The CHDIR command has another form. The subdirectory facility is a great organizational convenience, but you can easily get "lost" in the hierarchy and forget which is your current directory. The easiest way to find your current directory is to enter the CHDIR command without a path:

CHDIR

CHDIR d:

Current directory: C:\123\SALES
Commands: CD\123 (A)
CD .. (B)

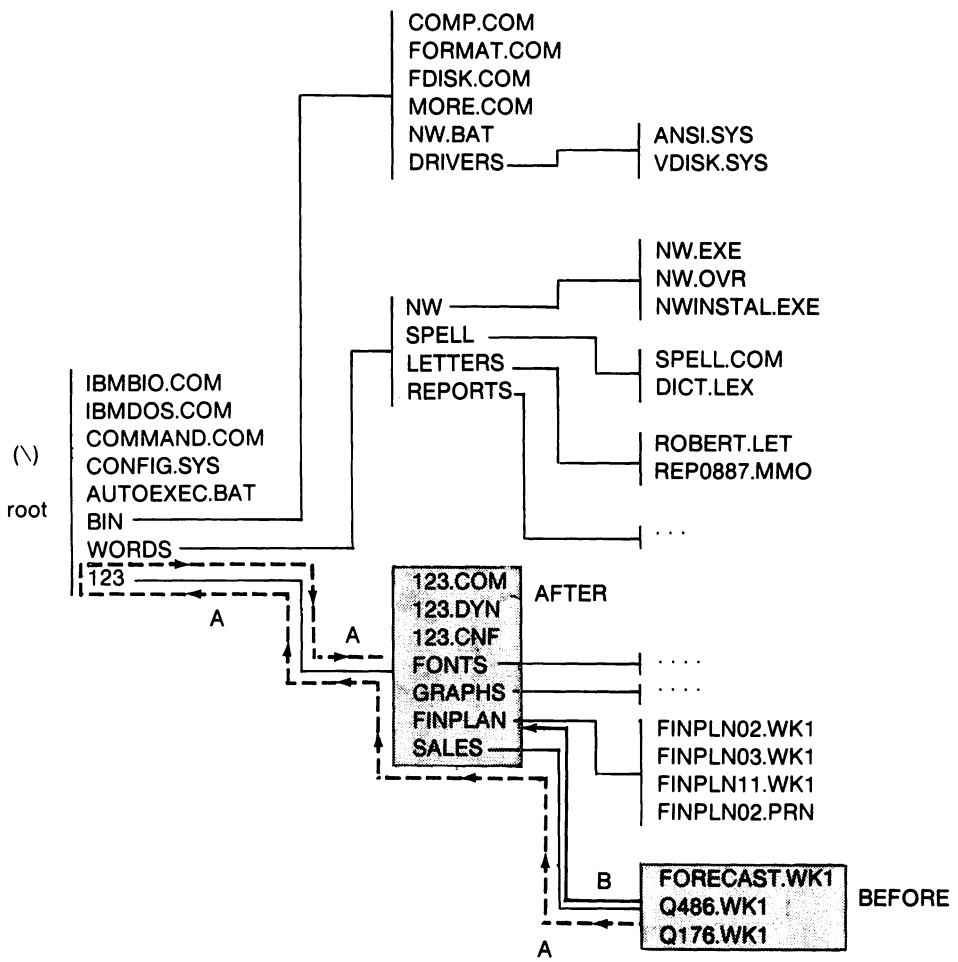


Fig. 5.7.

Remember that CD can be used in place of CHDIR. Because each disk drive has its own current directory, you can request DOS to show the current directory of any disk by specifying a disk drive name.

Current directory: C:\WORDS\LETTERS
Commands: CD\WORDS\REPORT (A)
CD\..\REPORT (B)

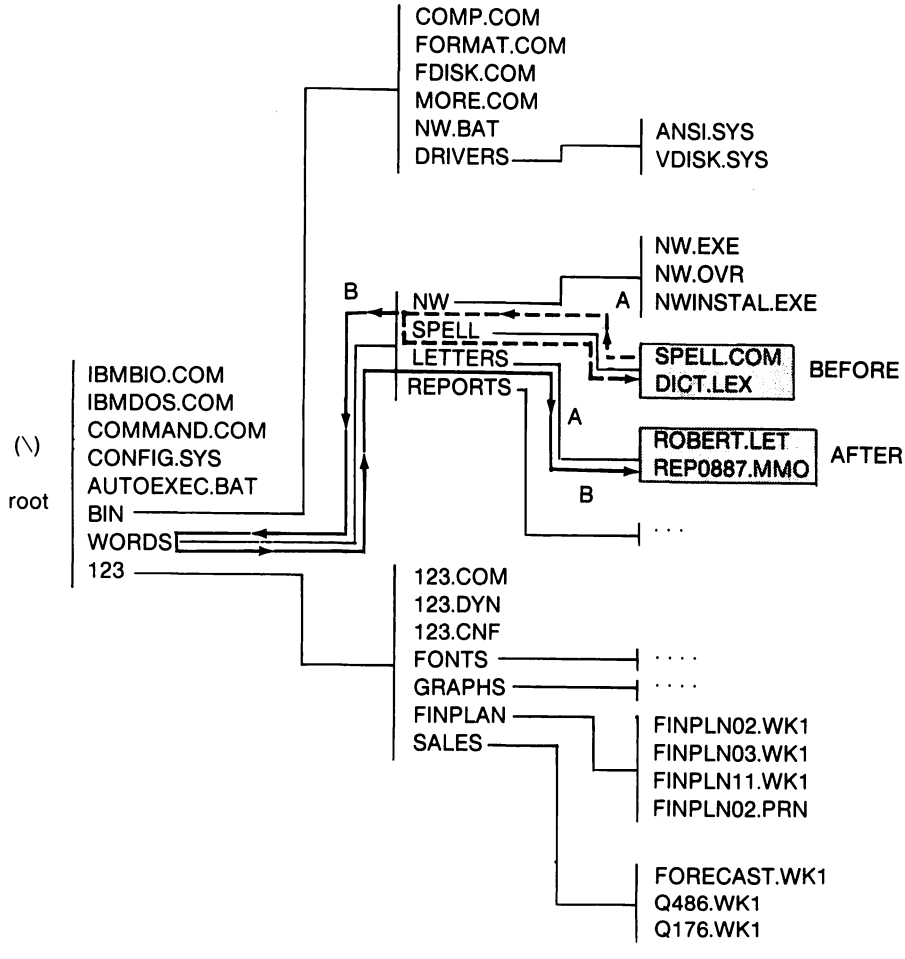


Fig. 5.8.

Deleting a Subdirectory

At times you may wish to remove a subdirectory. DOS has a command for this purpose: RMDIR or RD. However, DOS employs a fail-safe mechanism with this command. You can remove only

subdirectories that are empty, that do not hold any files or other subdirectories.

The command to erase, or *remove*, an empty subdirectory is one of the following:

RMDIR *d:path***directory_name**

RD *d:path***directory_name**

Only the last subdirectory in the path is deleted.

You cannot remove the current directory. You must be in the parent directory or another directory not lower in the chain of the directory to be removed.

DOS has one all-purpose error message for the misuse of the RMDIR command:

```
Invalid path, not directory
or directory not empty
```

This message means that (1) you provided an invalid path (an ill-formed or misspelled path name), (2) you provided an invalid directory (or the directory does not exist), or (3) you specified a directory that still contains some files other than the dot entries.

The error message reflects the second definition of the term *path*. This use of path denotes a complete name, either disk drive and directory names or drive, path, and file names. In this case, path means disk drive and directory names. Throughout this book, this use of path is avoided. However, you should be aware of this second meaning for path.

Occasionally, an apparently empty subdirectory cannot be removed. When you install some commercial programs, they create their own subdirectories and then place some hidden or read-only files in these subdirectories. A directory holding any files other than the dot and double-dot entries cannot be removed. The hidden or read-only files prevent you from removing a subdirectory. Fortunately, these programs usually provide an "uninstall" program that takes care of the problem.

You may also stumble when a problem occurs with your hard disk and files get altered. The "uninstall" program may fail to remove the hidden or read-only files. In this case, you will need a utility program to locate a hidden file or change a read-only file; then you can erase the file. These utilities are discussed in Chapter 10.

More about Paths

As already noted, path has several meanings. The first definition is the chain of directories to a certain file. The second definition of path is the complete file or directory name. These definitions come from the UNIX operating system, which views disk drives not as disk drives but as other directories in the hierarchy. Based on the second definition, each of the following is a path:

d:path\filename.ext
d:path\directory_name
d:path

The exact meaning of path is determined by the context. In some cases, path includes a disk drive and file name. In the DOS documentation, the hierarchical commands follow the third example, so path includes the final directory for your action (for making a directory or changing a directory).

This book uses the first definition of path: a chain of directories. Be aware that documentation usage does vary, and you may see the term *path* used as a chain of directories, or as a full file name.

The third definition for path is actually a DOS command. This definition is discussed in Chapter 6.

Summary

This chapter explains why files should be organized. The material introduces hierarchical directories and explains how they are made and removed. The text also discusses the concepts of the current directory and path names. The points to remember in this chapter are the following:

- Hierarchical directories help you organize and use files by placing related files together as a group and limiting the number of files DOS sees at one time.
- The path is the chain of subdirectories to a specific file. The path can also mean a full file specification.
- A full file name or file specification is a disk drive name followed by a path name followed by a file name.

- The path character (\) has two meanings. When used at the beginning of a path name, the backslash denotes that movement should start from the root directory. When used elsewhere, the path character separates directory names and the path name from the file name.
- Your system has one and only one current disk drive. Each disk drive has its own current directory.
- These DOS commands manage directories:

  MKDIR or MD makes a subdirectory.
  RMDIR or RD removes a subdirectory.
  CHDIR or CD changes the current directory.

The next chapter discusses several basic DOS commands and shows how these commands work with the hierarchical directory system.

Chapter 6

DOS Basics

To use hard disks effectively, you need a reasonable knowledge of DOS. This chapter reviews some of the basics of DOS and lays the foundation for the next two chapters, "Batch Files" and "Advanced Batch File Techniques." The discussion is based on the assumption that you understand the use of hierarchical directories and path names.

This book does not delve into all aspects of DOS. That's a subject for a different book. However, the text does review the fundamentals in order to ensure an equal starting point for all readers. If you're comfortable with DOS, just skim this chapter and move on. If you're in any way unsure, spend some time on this material.

Seeing the List of Files

To display the list of files and subdirectories on a disk, you use the DOS directory command, DIR. You probably have used this command frequently on floppy diskettes. Using DIR on a hard disk is no different, except that you use a path name more frequently.

The general form of the command is

DIR *d:path\filename.ext /P/W*

The *d:* is the letter of the drive you want; *path*\ is the path to the directory you want; and *filename.ext* is a valid file name. These three names are optional. If you need to see the directory of a disk other than the current disk drive, include the drive name. If you wish to see the directory of a subdirectory different from the current directory, give the path. If you want to see a list of some files in the directory, use a file name with one or more wild-card characters (* and ?). If you want to see a list of all files on the current disk drive or current directory, you can omit the three optional parameters altogether and simply type DIR.

You can use wild-card characters in a file name. An asterisk (*) matches any number of characters, and a question mark (?) matches only one character. You can use also uppercase or lowercase characters in all names, options, and commands (except for the FIND command, which is discussed in Chapter 10).

The DIR command has two *switches*, which are optional parts of the command that affect the way it works. The */P* switch causes the system to pause when the screen is full and wait for you to press a key. This is extremely handy when viewing a very long listing of files. The */W* switch produces a five-column-wide display of only the names of the files (see fig. 6.1). Without the */W* option, the listing displays the name of each file or subdirectory, the number of bytes occupied by each file, and the date and time of the file's creation or last modification (see fig. 6.2). At the beginning of the listing, DOS shows the disk volume name (if any) and the name of the directory. At the end DOS displays the total number of files and directories and the number of bytes of free space left on the disk.

```
C:\ -->dir \docs/w

 Volume in drive C is **DOS 3.1**
 Directory of  C:\DOCS

.          ..         LU      DOC   NUSQ03   DOC   ARC      DOC
OUTL   WR1 HDISK   TXT FUNDAY2       IBM32   DOC   PEGG3
PRECUR     TUNOTES    TUIEW          HD      WRK   CACHE
OUTLNEW WR1 FUNDAY    A1TEST  FW2
      18 File(s)   7555072 bytes free

C:\ -->
```

Fig. 6.1. *A five-column directory listing.*

```
 Volume in drive C is **DOS 3.1**
 Directory of  C:\DOCS

.            <DIR>      1-01-80  11:21p
..           <DIR>      1-01-80  11:21p
LW       DOC     6784   1-01-80   3:07p
NUSQ03   DOC    13034   7-12-85   3:07p
ARC      DOC    60168   1-31-86   3:07p
OUTL     WR1     6065   4-16-86  11:21p
HDISK    TXT     6272   8-16-86   1:18p
FUNDAY2          1664   5-10-86  11:21p
IBM32    DOC    11776   5-26-86  11:21p
PEGG3            7936   6-09-86  11:21p
PRECUR           1627   7-19-86  10:42a
TVNOTES          3848   7-20-86   3:15p
TVIEW            3968   7-20-86   3:30p
HD       WRK    22277   7-27-86   8:29a
CACHE            2191   7-27-86  11:30a
OUTLNEW  WR1     9353   8-17-86   7:50a
FUNDAY           2526   8-09-86   4:14p
A1TEST   FW2      352   8-13-86  12:38p
      18 File(s)   7557120 bytes free

C:\ -->
```

Fig. 6.2. *A directory listing without the /W* option.

For all DOS commands, spaces are significant except before and between the switches. In the DIR command, for instance, you must leave at least one space between *DIR* and the start of the full file name. More spaces can be used, but most people prefer to enter as few characters as possible. A space is not needed between the full file name and the switch character. DOS separates the file name from the switches with the switch character, the slash (/). For most commands, spaces are not needed between switches. You must be careful also not to insert a space where none should be. DOS is not highly intelligent and will misinterpret your commands.

With the command

DIR C:\ /P

DOS displays a list of all the files *and* subdirectories in the root directory of drive C. When the screen fills, DIR pauses and waits for you to press a key before the listing continues. Note that you explicitly asked for DOS to give you a listing for the root directory of drive C. Because you did not give a file name, all files (and subdirectories, which "look" like files to DOS when you are requesting a directory) are included implicitly.

To get a listing of the current directory on the current disk drive (and have the display pause at each screenful), you type

DIR /P

Because no disk drive is specified, the current disk drive is used. No path is specified, so the current directory is used.

Getting a listing of files for a different directory is slightly more difficult. You must specify a path to help DOS maneuver through the directory chain (see fig. 6.3). For example, to see the list of files and subdirectories in the directory C:\WORDS\REPORTS, you use

DIR C:\WORDS\REPORTS /P

Current directory: Any
Command: DIR C:\WORDS\REPORTS/P

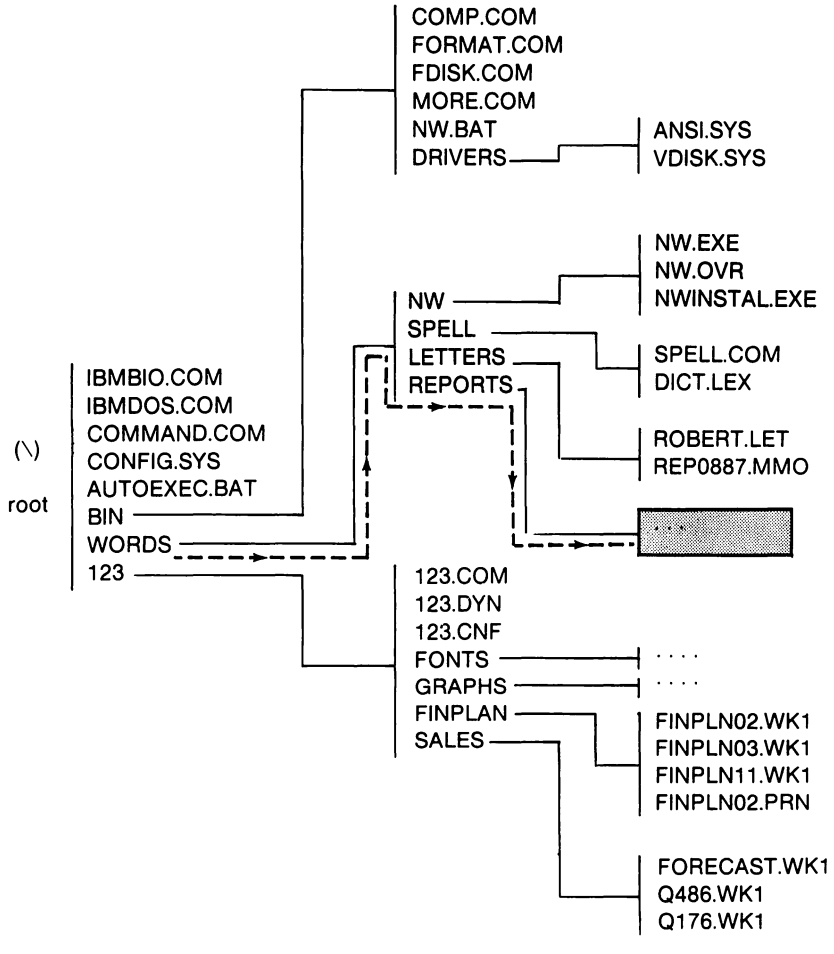


Fig. 6.3.

If your current directory is WORDS and the current disk drive is C (fig. 6.4), you achieve the same result just by typing

DIR REPORTS /P

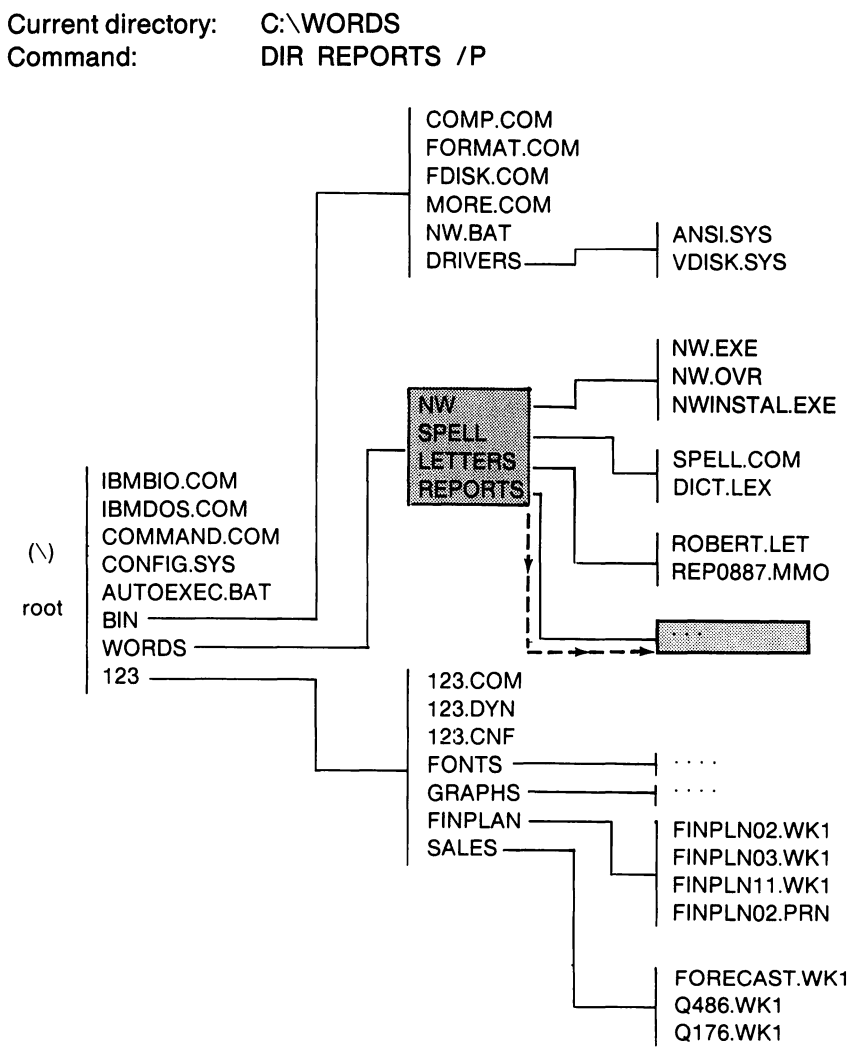


Fig. 6.4.

You may notice that I like using the */P* option; otherwise, a long list of files can scroll off the top of my screen before I remember to press Ctrl-NumLock, a manual way of stopping *any* screen listing. With this option, pressing any key "unfreezes" the screen.

DIR is a harmless command. You do not change or alter any disks or files when you get a directory. To practice using path names, use DIR to test different combinations of disk drive, path, and file names from different directories. After you have mastered using the DIR command on different subdirectories, the remaining DOS commands are easier to use.

Copying Files

The COPY command is used to copy files between disk drives and subdirectories. The simplified command for COPY is

COPY *ds:paths\filenames.exts dd:pathd\filenamed.extd /V*

COPY can accept up to two complete file name specifications. The first complete file name specifies the file or files to be copied, the *source*. The second file name is the *destination*, which states where the file will go and what new name the file will have. In the preceding example, each part of the source-file name ends with an *s*, and each part of the destination-file name ends with a *d*. When several file names are possible, a character or number is added to distinguish the different names.

Because COPY accepts wild-card characters in either file name, a single file name can specify more than one file.

COPY works a little differently in DOS V2 than it does in later versions of DOS. In Version 2, you must provide at least the first (source) *filename*. The remaining parts are optional. In Version 3 and later, if you give a path in the source name you can omit the file name. In this case, the wild-card name *.* is assumed and all files are copied. How much of the COPY command you must include depends on what files you wish to copy and where the copied files are placed.

DOS does not allow two files to have the same name in the same directory. If you are copying files within your current directory, you must change their names (give at least the second *filename.ext*). If you are copying files between disk drives and directories, you must include both. Otherwise DOS "assumes" that you mean the current disk drive and current directory.

Some examples help illustrate the minimum requirements. Suppose the current disk and directory designation is C:\WORDS\REPORTS. To copy FINPLN02.PRN to the current directory, the full command is

COPY C:\123\FINPLAN\FINPLN02.PRN C:\WORDS\REPORTS\FINPLN02.PRN

Now drop the unnecessary parts. Because the current disk drive is drive C, the disk drive names can be dropped (see fig. 6.5). That changes the command to

COPY \123\FINPLAN\FINPLN02.PRN \WORDS\REPORTS\FINPLN02.PRN

Current directory: C:\WORDS\REPORTS
Command: COPY C:\123\FINPLAN\FINPLN02.PRN
C:\WORDS\REPORTS\FINPLN02.PRN

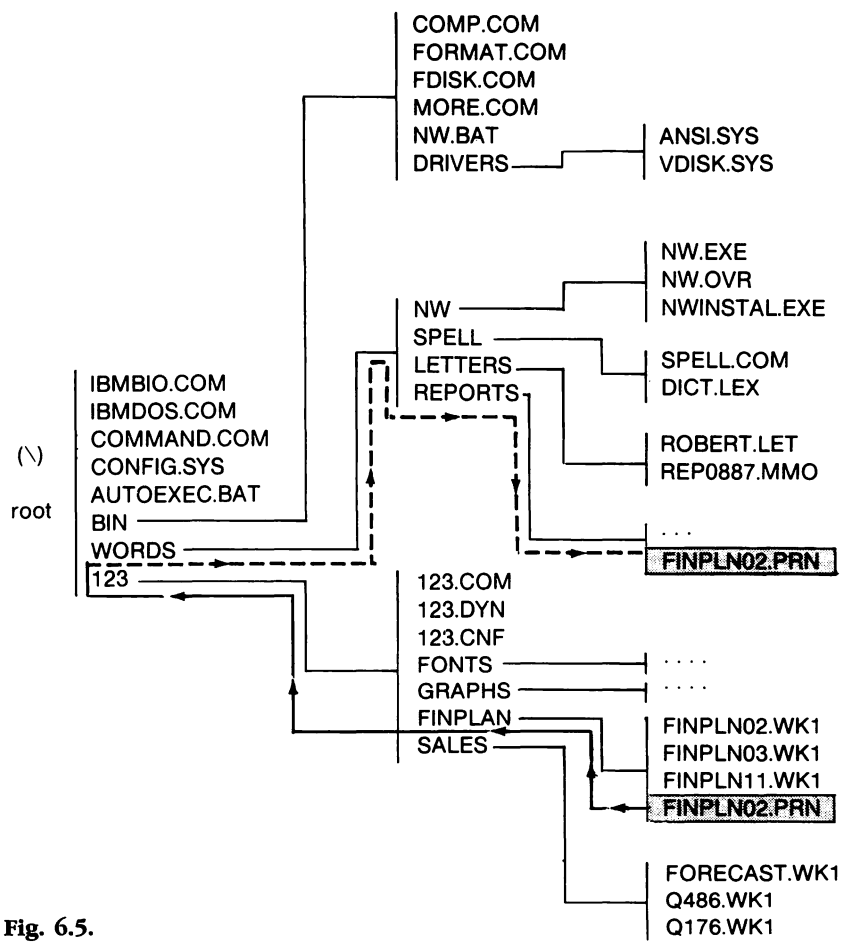


Fig. 6.5.

You wish to have the copied file keep the same as the source file, so you can omit the second file name. DOS correctly assumes that you do not wish to alter the copied file's name. The command becomes

COPY \123\FINPLAN\FINPLN02.PRN \WORDS\REPORTS

Finally, because you are copying the file into your current directory, you drop the path name for the destination (see fig. 6.6). The command becomes

COPY \123\FINPLAN\FINPLN02.PRN

Current directory: C:\WORDS\REPORTS
Command: COPY \123\FINPLAN\FINPLN02.PRN

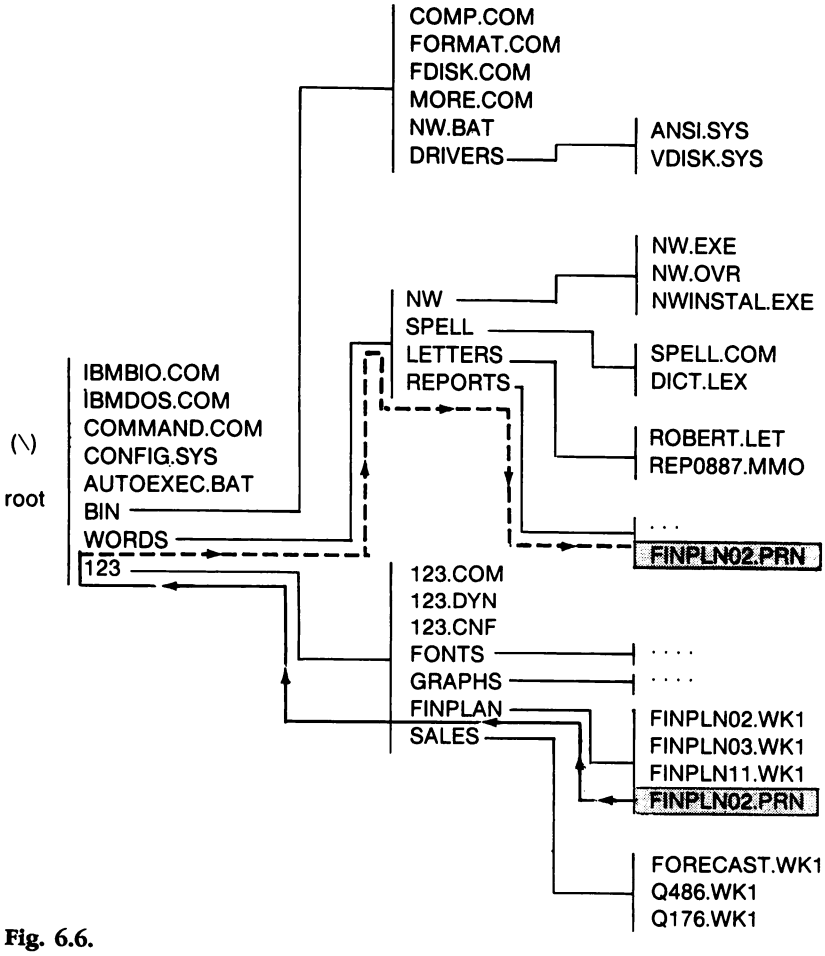


Fig. 6.6.

This command is the shortest and simplest version you can give to DOS to copy FINPLN02.PRN into your current directory.

Next, assume that your current directory is C:\123\FINPLN (see fig. 6.7). To place FINPLN02.PRN in your REPORTS subdirectory, you use the command

COPY FINPLN02.PRN \WORDS\REPORTS

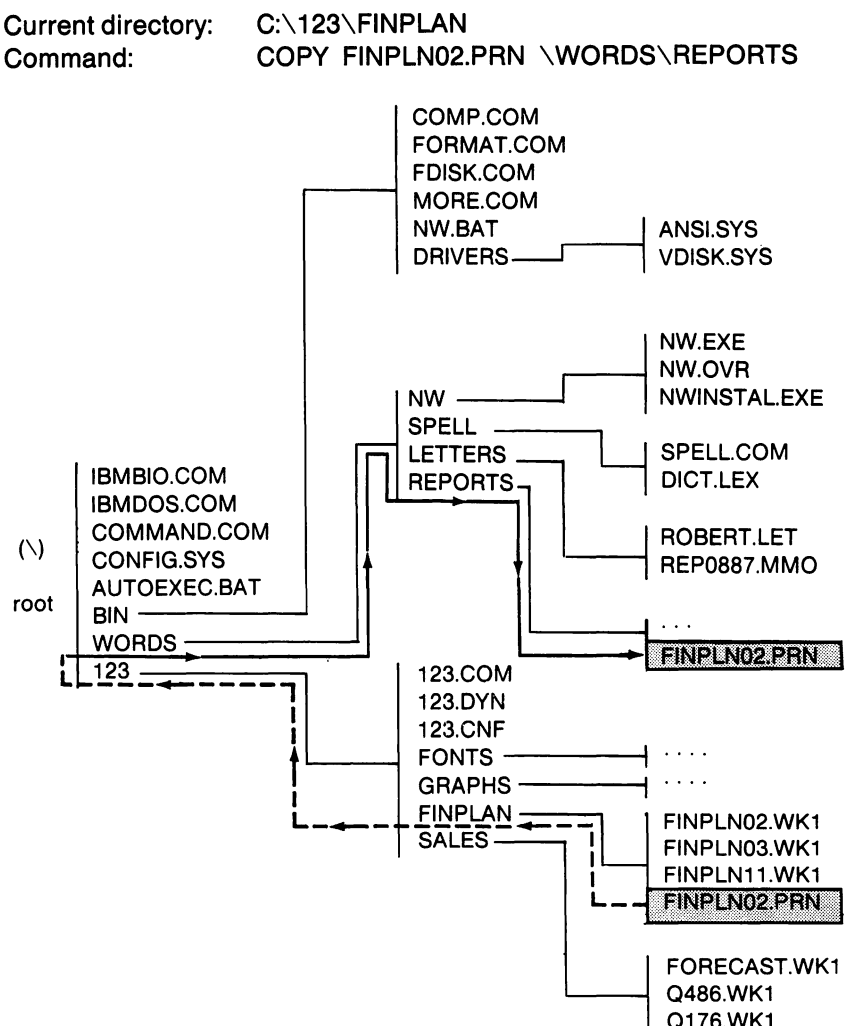


Fig. 6.7.

Because you stay on the current disk drive, no drive name is needed. The file is in your current directory; therefore, you need not specify the path name for the source. You must, however, specify part of the destination. Because the subdirectory where the copy will be placed is not the current directory, you must specify the destination path.

What if neither your source nor your destination is the current directory? Again assume that your current directory is C:\123 (see fig. 6.8). To copy the same file, you use

COPY FINPLAN\FINPLN02.PRN \WORDS\REPORTS

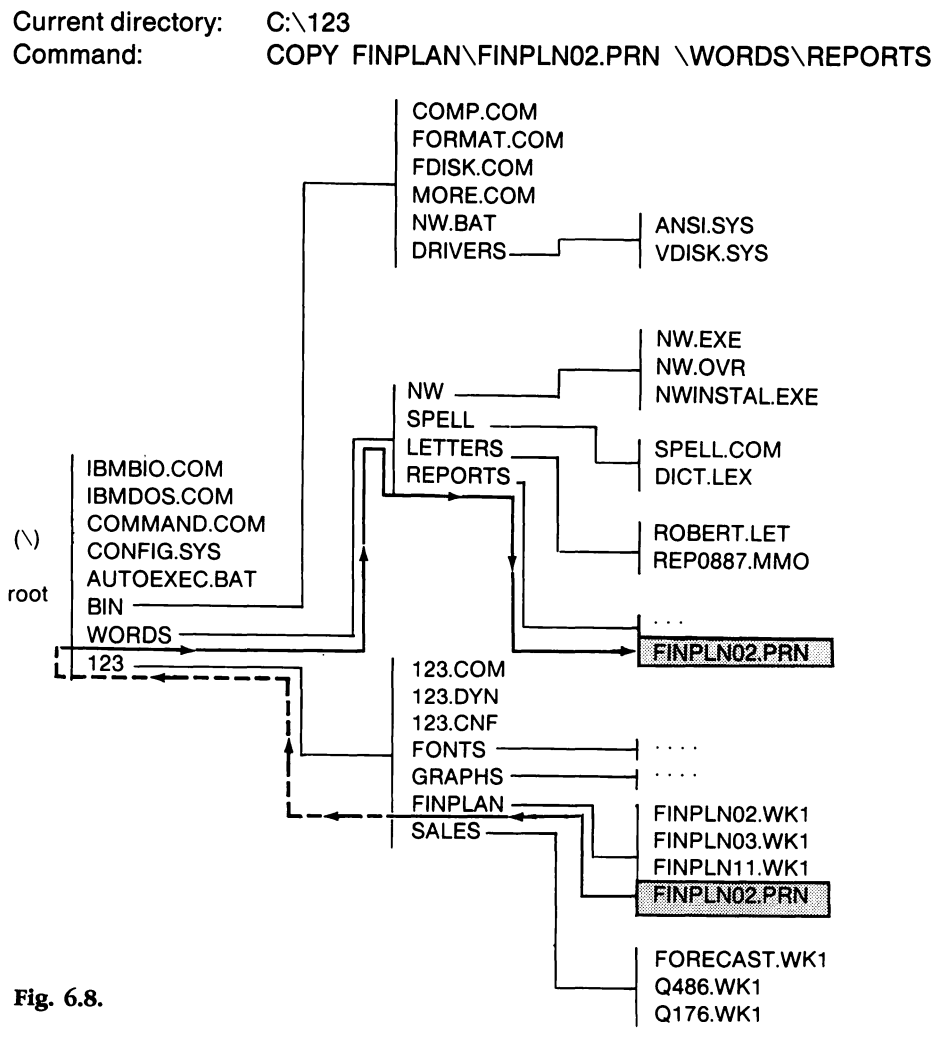


Fig. 6.8.

The rationale for including both path names is that both the source and the destination are directories other than the current directory. Because FINPLN is a subdirectory of 123, your current directory, you direct DOS to this directory by giving FINPLN as the path. You need not send DOS back to the root directory, so a source path of \123\FINPLN is not necessary. Because the destination path is one level up and two levels down from your current directory, giving \WORDS\REPORTS is the shortest path name you can use.

Giving a more complete file name than is absolutely necessary causes no harm, but one goal of this book is to increase your efficiency and comfort in using DOS. Part of achieving this goal is to make typing commands easier by omitting unneeded portions of the command.

The DOS VERIFY Command and COPY /V

All of COPY's options need not be discussed, but the */V* or *verify* switch and DOS's VERIFY command are important and should be discussed at this point.

Some people refer to the */V* switch and the VERIFY command as the *very* long options. When you give the */V* option or turn on VERIFY, you request that DOS check (verify) that each new file is readable. The extra activity while copying (or using DOS) takes more time—about twice as much time.

The question is "Should you bother with verification?" The answer depends on your particular daily use of the COPY command. If you use COPY for backup purposes, the */V* switch is highly recommended. The backup copy must be right. Unless you use the */V* option, you may never know that a copied file is corrupt (unusable) until your original "goes down the tubes." That time is definitely the most undesirable time to experience the "thrill of discovery." If the copy is poor, you want to know *immediately*, at the time you make the copy.

You can invoke verification in two different ways. You can call for it directly in the COPY command itself or as a separate command. The first method looks like this:

COPY A:CHAPT.* C:\DOCS /V

In this example, every file with a root name of CHAPT and any (or no) extension is copied from drive A to the DOCS subdirectory of drive C, and each copy is verified.

The second method invokes the VERIFY and COPY commands separately:

```
VERIFY ON
COPY A:CHAPT.* C:\DOCS
COPY A:BOOK?.* C:\DOCS
COPY A:EDIT.* C:\DOCS
COPY B:*.WR1 C:\SYMPH
VERIFY OFF
```

The first and last commands show the syntax of VERIFY. The first command turns on the verification mode. The last VERIFY command turns off verification. Any DOS disk activity between the two commands (including but not limited to COPY commands) is subject to the DOS verification process. This method is a bit handier when you are doing a series of COPY commands or running commands other than COPY. When VERIFY is on, the */V* option with COPY is unnecessary but harmless.

You can find out how VERIFY is set by typing the command without ON or OFF. DOS displays a message stating that VERIFY is either on or off.

VERIFY affects all DOS operations that write to a disk—not just DOS commands. Verification slows the operation of any programs that create and write files. All DOS disk-write operations take about twice as long. Any operation to read information from a disk, however, is unaffected.

COMP, an Alternative to Verification

Verification is not absolute insurance that a file has been copied correctly. When verification is turned on (VERIFY ON or COPY /V), DOS checks that (1) the information has been recorded properly, (2) the internal housekeeping information is properly recorded, and (3) the file can be successfully read from the disk. Verification only checks that the copy can be read but does not check that the information in the copy precisely matches the original.

Although the probability is slight, some problem with the computer might conceivably cause DOS to write a successful copy of a file with the contents not matching the original file. As long as DOS can read the data, the system does not signal a VERIFY error.

Although this misguidance of DOS is not an everyday occurrence, do you want to take a chance that an important copy is corrupt?

Fortunately, DOS has an underutilized command called COMP, which can ensure that your files do match. COMP *compares* the contents of two files or two sets of files, byte by byte. If the files do not match precisely, COMP informs you of the difference. COMP performs what many people think VERIFY does.

Most users religiously execute the COMP command immediately after any important copying. If you follow this practice, you can skip the COPY */V* option and routinely use COMP after each COPY.

The syntax for COMP is

*dc:pathc***COMP** *ds:paths**filenames.exts dd:pathd**filenamed.extd*

Notice that the syntax of COMP is different from COPY or DIR. COMP is a disk-based external DOS command. You must load COMP into memory before you can run the program. If COMP is not on the current disk drive or in the current directory, you include *dc:* (the disk drive name) or *pathc*\ for the path to the directory holding the command, as needed. Note the added *c* in the syntax description to designate the drive and path for the command. The *pathc*\ designation is effective only for DOS V3 and later. DOS V2 users cannot run a command that is not in the current directory. The first file specification designates the source files for the comparison. The second file specification specifies the destination files. DOS uses the terms *primary* and *secondary*, respectively, for the source and destination file specifications.

You include disk drive and path names as needed. You may use wild cards with the COMP command; if you don't include a file name, DOS assumes *.*. If DOS assumes *.* or you use any wild-card character, DOS compares files with names in the source and destination set that match *exactly*. In other words, when using wild-card file names, DOS compares only files with the same file name in both sets. If DOS finds a file name in the source set that has no counterpart in the destination, DOS displays a warning that it cannot find the counterpart in the destination set.

If you omit all parts of the source and destination, COMP prompts you for each name. COMP is one of the few DOS commands that does not require that all information be typed on the same line with the command.

After each pair of files is examined, COMP reports the results. If everything is correct, you see the message

```
Files compare ok
```

Occasionally, even if the files are the same, you may get an additional message just before the preceding message:

```
EOF mark not found
```

EOF mark stands for *end-of-file marker.* This message should not concern you as long as the files being copied are program files, not text files. The end-of-file marker is a code that programs place in a text file to indicate the end of the file. If you compare two text files and get the EOF error message without the OK, you should double-check the text file copy to see that the file copied correctly.

When COMP finds a mismatch, the computer beeps and displays a warning message, something like this:

```
Compare error at offset XXXX
File 1 = YY
File 2=  ZZ
```

XXXX, YY, ZZ are hexadecimal (base 16) numbers indicating, respectively, the *offset,* or the number of bytes into the file where the discrepancy occurred, and the actual values that differed in each file. COMP displays up to ten mismatches, and then skips to the next two files for comparison. Finding ten mismatches is sufficient evidence that the files are indeed different.

COMP does not bother comparing files of different sizes listed in the directory of the disks. COMP simply displays the message

```
Files are different sizes
```

COMP then goes to the next two files.

When should you use COPY and COMP? Generally, use the COPY-COMP team when copying files in which you have no room for error. I am more biased toward COPY-COMP when I am copying to a floppy diskette. Floppy disk drives do not have the sophisticated self-correcting electronics that hard disk drives have. You will have more errors when copying to a floppy diskette than when copying to the hard disk. The chance of creating a corrupt copy is exceedingly small; but when no error can be tolerated, use COMP after COPY.

Shortcut Commands

Because both COPY and COMP are four-letter words, you can use the DOS editing keys for a shortcut when you enter COMP after a COPY operation. For example, suppose that you give the following command to copy files:

COPY A:CHAPT.* C:\DOCS

When the COPY command is finished, type

COMP

Then press the F3 function key. This key repeats what remains of the preceding line typed into DOS. Because COMP and COPY have the same number of characters, pressing the F3 function key repeats the rest of the COPY command line (A:CHAPT.* C:\DOCS), which is exactly what you want. You can do this with each COPY command.

DOS V2.1 COPY Fluke

The DOS V2.1 COPY command has an undocumented feature (read "bug"). You may have used COPY for years and not noticed the problem because the circumstances that give rise to it are uncommon. If you attempt to copy all files from a directory (using *.*) and the number exceeds 256 files, every 256th file is skipped. No message appears. COPY simply gets befuddled and skips every 256th file. This problem sounds oddball, and it is not exactly a situation you would ordinarily anticipate. Copying more than 256 files at a clip is not common practice. However, hard disk owners could stumble over this flaw.

The sneaky part of this bug is the fact that DOS doesn't indicate any problem but just lets the doomed file slip into the netherworld. When copying a large number of files, you seldom check that each file has been copied. Therefore, this bug could remain unnoticed until you need that 256th file.

What defensive measures should you take? You could count the files in the source and destination by using the DIR command. Or, if you use COMP after a COPY command, COMP automatically complains about the missing file. One other choice is to upgrade to DOS V3 or later, which also cures other problems in DOS V2 and gives you some helpful commands unique to DOS V3 and later.

Erasing Unwanted Files: DEL and ERASE

To remove an unwanted file, use the DEL (delete) or ERASE command. The two are built into DOS and are synonyms, so you can interchange the two command names. The syntaxes are the same:

DEL *d:path\filename.ext*

ERASE *d:path\filename.ext*

The *d:* is the name of the disk drive holding the file(s) to be removed; *path*\ is the directory path to the files to be removed; and *filename.ext* is the name of the file(s) you wish to remove (see fig. 6.9). ERASE and DEL accept wild-card characters, so a single file name can remove many files.

Current directory: Any
Commands: DEL C:\123\FINPLAN\FINPLN02.WK1
ERASE C:\123\FINPLAN\FINPLN02.WK1

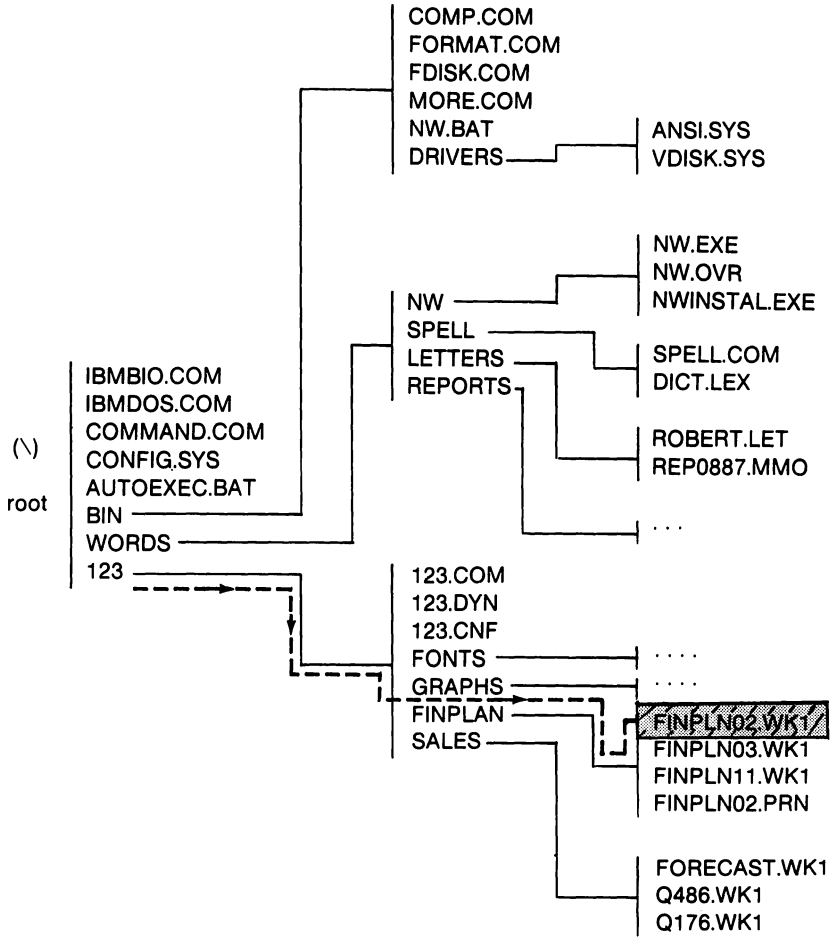


Fig. 6.9.

To erase the file FINPLN02.WK1, for example, you type

DEL C:\123\FINPLAN\FINPLN02.WK1

DOS V2 reacts slightly differently from DOS V3 and later. In DOS V2, ERASE or DEL insists on a file name. DOS V3 allows you to omit the file name if you give a path name. However, if you omit the file name, DOS uses *.*, and all files in the specified directory are erased.

DOS is cautious when you use a file name of *.*, implicitly or explicitly. DOS asks

```
Are you sure (Y/N)?
```

If you wish to erase all files in the directory, press **Y**. If you are unsure or have given the wrong file name, press **N**.

When you are erasing files with a wild-card name, watch your typing and be sure that the file name you give does not include any files you do not wish to erase. If in doubt, first specify the file name with the DIR command. Then check the list for any files you do not wish to erase. If any show up, you must use a different file name.

Once a file is deleted, it cannot be recovered using DOS-provided programs. You will need some outside tools to recover erased files. These programs are discussed in Chapter 10.

Finding Files and Subdirectories: TREE

One problem with having an extensive hierarchical-directory system is remembering the subdirectories and the files that are located in each subdirectory. DOS provides a helpful tool: the TREE command.

The syntax for TREE is

*dc:pathc***TREE** *d: /F*

TREE is an external command; it must be loaded from a disk and run, so you need to use the drive and path name for the TREE command. TREE accepts only one *argument* (additional information after the command) and a single switch. The *d:* is the optional name of the disk drive for TREE to analyze. If you omit *d:*, TREE analyzes the current disk drive. The */F* switch directs TREE to print not just subdirectory names but also file names. Remember that if you omit the */F*, files in each directory are not shown.

Better tools to help find files and subdirectories are discussed in Chapters 10 and 11.

Redirecting Input and Output

DOS V2 and later versions offer the convenient shortcut of I/O redirection. The input to a program can be received from a device or file and the output from a program can go to a device or file. The four redirection symbols and their meanings are given in table 6.1.

Table 6.1
Redirection Symbols

Symbol	*Meaning*
<	Redirects a program's input from the keyboard to a disk file or other device
>	Redirects a program's output from the screen to a disk file or other device, such as a printer
>>	Same as >, but appends the program output to the end of a file
|	Redirects the output of the program named on the left side of the bar to the input of the program given on the right side of the bar

Three of the four redirection symbols are often used to trick programs into getting their input from a disk drive or sending their output to a disk file or the printer. For example, consider the command

TREE >DISKTREE

This command takes the text that the TREE program normally sends to the video screen and captures that text in a disk file named DISKTREE. You can then print or edit the file.

The most popular use of redirection is to send a program's output to the printer. The following commands illustrate this use:

DIR >PRN
TREE >PRN

You do need a word of caution about using the > symbol: Suppose you use > to redirect the output of a program into a disk file that

already exists. DOS first deletes the named file and then creates the file to hold the output. In the example of redirecting TREE's output to DISKTREE, DOS erases from the current directory on the current disk any file with the name DISKTREE. DOS then creates a new DISKTREE to hold the output of TREE. If you inadvertently give the wrong name with >, DOS destroys the file.

The symbol >> does not cause the same problem. When you use >>, DOS adds (appends) to an established file instead of destroying an old file and creating a new file. If the named file does not exist, DOS creates the file.

I/O redirection is a valuable tool for any DOS user. Become acquainted with using >PRN to get a hard copy of a program's output. Using TREE >PRN, for example, provides a helpful road map for your hard disk. Printouts from DIR >PRN can be stored with a diskette or near the computer to help users locate needed files. More uses of I/O redirection are given throughout in this book.

Using Common Programs with the PATH Command

The last chapter explains that path has three meanings. The first two refer to directory path names and full file names. The third is a handy feature built into DOS V2 and later: the PATH command.

As mentioned, if a program is not in your current directory, you must specify the path to that program. DOS V2 users do not have this option. The program must be in the current directory or the program cannot be run.

The PATH command is a saving grace. When PATH is used, DOS automatically searches one or more different directories for your program. The PATH command is best used in the AUTOEXEC.BAT file. Including PATH in AUTOEXEC.BAT causes DOS automatically to establish the search path for your programs each time DOS is started. (See Chapters 7 and 8 for information about batch files.)

The syntax for PATH is

PATH *d1:path1;d2:path2;d3:path3;* . . .

The *d*'s are the optional disk drive names of the paths. Each *path* is the path to the subdirectories that DOS should search when looking for a program to run.

When you run any executable file—a program, an external command, or a batch file—without giving a path name, DOS looks in the current directory for the executable file. If DOS cannot find the file, DOS then searches each directory in the sequence specified by the PATH command. If the file is not in the directory specified by *d1:path1*, DOS searches *d2:path2*, then *d3:path3*, and so on, until the file is found or DOS runs out of paths for the PATH.

The PATH command allows you to place executable files in one or more established subdirectories, then invoke a file without needing to type a drive and path name each time. PATH offers a powerful shortcut for running programs.

Users often place frequently used programs in one or two directories. For instance, most people place their DOS programs and other command programs in a directory called either C:\BIN or C:\DOS. Suppose that you set the PATH to

PATH C:\BIN

or

PATH C:\DOS

When DOS fails to find the needed program in your current directory, DOS automatically searches the BIN or DOS subdirectory and finds and runs the program.

Some programs ingraciously require certain files in the root directory, so a more frequent PATH command is

PATH C:\BIN;C:

or

PATH C:\DOS;C:

These commands include the root directory of drive C in the search (see fig. 6.10). DOS does not automatically look at all subdirectories in a specified path, only the last subdirectory in the given paths. C:\BIN or C:\DOS alone does not tell DOS to search the root directory also.

The sets of drive and path names are separated by semicolons (;). Except for the space between the word PATH and the first path, no other spaces are allowed. DOS searches each path in the order in which the paths are specified. The directory holding your most frequently used program should be specified first, the second most frequent directory second, and so on.

Current directory: N/A
Command: PATH C:\BIN;C:\

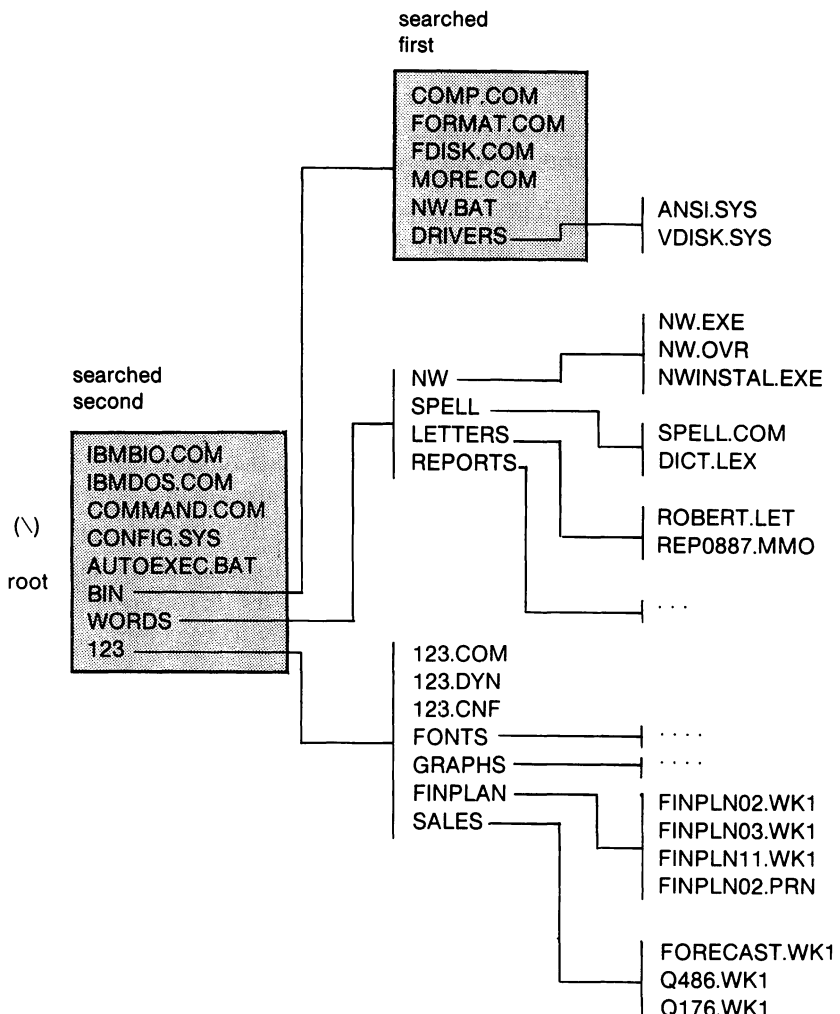


Fig. 6.10.

PATH has one limitation. PATH only tricks DOS into finding programs, commands, and batch files not in the current directory. PATH does not trick programs in their search for data files or program overlays (additional parts of a program located in separate files and loaded in only as needed). If your programs do not allow path names, you'll need another program to trick your programs (see Chapters 9 and 12).

Summary

This chapter covers some frequently used DOS commands. Each command uses path names and allows you to affect files that are not located in the current directory. Although this chapter does not fully cover each command, the frequent uses of the commands are shown. The points to remember are

- The COPY command makes duplicates of files.
- The */V* switch of COPY and the VERIFY command enable DOS to check that the copies are correctly made, but these commands do not check that the contents of the files are identical.
- The COMP command compares two files byte by byte to ensure that the files are exactly the same.
- The DEL and ERASE commands remove unwanted files.
- The TREE command displays the subdirectory hierachy of your disk and can include the files held in each subdirectory.
- I/O redirection can be used to redirect the input and output of programs. The most frequent use is to direct a program's output to the printer by adding >PRN to the command.
- The PATH command allows DOS automatically to search other directories for programs, commands, and batch files.

The next chapter introduces batch files, a method to automate common DOS commands.

Chapter 7

Batch Files

In Chapter 4 you learned to create and use AUTOEXEC.BAT, a special kind of *batch* file that DOS executes automatically when you start the system. A batch file provides a shortcut for executing one or many DOS commands by typing a single file name. DOS executes each line in the file as if you had entered the line from the keyboard.

Batch files are extremely convenient and useful tools. Repetitive or long instructions can be automated. The chance of mistyping a command is reduced, and long tasks can be started and left to run unattended.

Many software manufacturers include batch files with their programs. These batch files ease the procedure of installing the program and configuring the applications to your operating system. Rather than giving users a complex series of directions telling them to create different subdirectories and copy specific files, the manufacturer prescribes just one step: run the batch file. The procedure is quick and painless.

Batch files contain ASCII (humanly readable) text. This chapter explains the batch subcommands and gives examples of batch files you can use. You can create batch files by using the COPY CON technique, any word processor capable of creating text files (most

can), or a text editor (including the DOS EDLIN editor). COPY CON is not recommended for the longer examples because of the inconvenience in correcting typing errors.

If you use a word processor, use the programmer, or nondocument, mode. The normal mode of many word processors stores the characters you type in a code that DOS may not understand.

The batch files given in this chapter are typed in all capital letters. DOS accepts lowercase letters just as easily except in special cases that are pointed out in the text.

Remember that all batch files have one thing in common: the extension is BAT. No other extension is allowed.

A Simple Example—Starting a Program with a Batch File

Chapter 5, on setting up directories, mentions that the sequence of directory commands leading up to the running of a particular program can be automated by placing them in a batch file. As a simple example, suppose that you regularly start a program called APPLIQUE from a particular directory. You call up the program, copy some files after you exit from the program, and then jump back to the root directory. You can set up a batch file containing the following lines (assume that the hard disk is drive C):

```
C:
CD \APPDATA
APPLIQUE
COPY *.DAT B:
CD \
```

Suppose that this batch file is named A.BAT. When you want DOS to execute the series of commands in the file, you type just

```
A
```

Then press Enter. No extension is needed.

When you type **A** and press Enter, DOS looks in the current directory for a file named A.COM, A.EXE, or A.BAT, in that order. Providing you do not have A.COM or A.EXE in your directory, DOS runs the batch file A.BAT. (As mentioned in preceding chapters, you should never use a batch file name that duplicates the name of a

COM or EXE file.) As the batch file runs, you can see the commands displayed on your screen and then executed one by one.

If you have on this directory a subdirectory or file called A.TXT, DOS displays

```
Bad command or file name
```

Remember that DOS executes only files with extensions of COM, EXE, and BAT. If you use an extension of TXT (or any extension but these three), DOS cannot find a file the program can execute. DOS therefore displays the error message.

Batch File Commands

Any valid DOS command can be used in a batch file. DOS also has a set of commands specifically for use in batch files: the batch subcommands. The seven batch subcommands are REM, ECHO, PAUSE, GOTO, IF, FOR..IN..DO, and SHIFT. Table 7.1 lists each command with a brief description of its purpose.

Table 7.1
Batch Subcommands for DOS V3

ECHO	Turns on or off the display of batch commands and displays a message on the screen
FOR..IN..DO	Allows the use of the same batch command for several files
GOTO	Jumps to the line after a label in the batch file
IF	Allows conditional execution of a command
SHIFT	Shifts the command line parameters one parameter to the left
PAUSE	Halts processing until a key is pressed and optionally displays a message
REM	Displays a message on the screen

Displaying a Message: REM

Often a batch file must prompt a user for some action, such as placing a particular disk in the appropriate disk drive. You can build prompts into your batch files with subcommands. One batch

subcommand for prompts is REM, short for *remark*. Whatever you type, including the starting REM, is displayed on the screen, but no action is taken. Here's the beginning of a sample batch file demonstrating REM commands:

```
REM To run this program, make sure the disk containing
REM BASICA is in drive A and the disk labeled
REM CONTRACTS is in drive B.
REM ..............................................................
REM (If you need to exit at this time, hold down the
REM Ctrl key and press the Break key. When the system
REM asks you whether you wish to "Terminate batch job (Y/N)?"
REM press Y. The batch file will stop.)
```

When this batch file runs, DOS displays each remark line on the screen. Nothing on the line is "executed" by DOS. You see on the screen the word *REM* itself and the full line that follows the word *REM*.

Because REM's display of messages is not "tidy," I suggest using the batch subcommand ECHO.

Controlling a Displayed Message: ECHO

As you become more experienced with batch files, you will often create files for others to use. In this case, you need to "clean up" the screen and display only needed information. One aspect of the ECHO subcommand handles this task. Another aspect of ECHO gives cleaner batch file messages.

ECHO OFF

The following example contains the same information given in the previous batch file. However, this file uses one form of ECHO to turn off the messages and changes all REM commands to ECHO statements. Here's the new version of the file:

```
ECHO OFF
ECHO To run this program, make sure the disk containing
ECHO BASICA is in drive A and the disk labeled
ECHO CONTRACTS is in drive B.
ECHO ..............................................................
ECHO (If you need to exit at this time, hold down the
```

ECHO Ctrl key and press the Break key. When the system ECHO asks you whether you wish to "Terminate batch job (Y/N)?" ECHO press Y. The batch file will stop.)

The first command, ECHO OFF, is the "noise suppressor." ECHO OFF suppresses the "echoing" (displaying) of any batch commands on the screen as that command is executed. However, ECHO OFF does not turn off messages displayed by another ECHO command. ECHO OFF affects REM statements but not other ECHO statements. Running the preceding chunk of a batch file produces the following display:

```
C>ECHO OFF
To run this program, make sure the disk containing
BASICA is in drive A and the disk labeled
CONTRACTS is in drive B.
.................................................
(If you need to exit at this time, hold down the
Ctrl key and press the Break key. When the system
asks you whether you wish to "Terminate batch job (Y/N)?"
press Y. The batch file will stop. )
```

To see the differences between the two commands, enter the batch file using REM and execute the file. Then enter the version using ECHO (or edit the old batch file) and execute this file. Using ECHO OFF and an ECHO message gives neat and informative messages while hiding the operational clutter that can confuse novice users or distract frequent users.

Both ECHO and REM statements are useful in a batch file. You can use ECHO OFF to suppress messages and use REM statements as comments in the batch file. The REMs are suppressed and do not clutter the screen, but the comments serve as documentation when you or others need to modify the file later.

ECHO ON

ECHO ON is the counterpart to ECHO OFF. When ECHO is turned ON, every batch file line, including REM statements, is displayed. When you start any batch file, DOS runs the batch file as if ECHO ON were the first line of the file. A later section describes how batch files can run other batch files. The examples in that section show the effect of starting a batch file with an implicit ECHO ON.

The implicit ECHO ON of batch files differs between DOS V2 and later versions. With DOS V2, when the first batch file runs another batch file, ECHO is automatically turned back on, regardless of whether ECHO was turned off by the preceding batch file. DOS V2 batch files must have an ECHO OFF statement in every batch file to keep ECHO turned off. With DOS V3, once ECHO is turned off, it stays off until the batch files are finished or an ECHO ON command is given. All subsequent batch files are "silent." This property of ECHO is important when batch files run each other. This aspect, however, does not apply when you manually run more than one batch file. Once any batch file returns to DOS, ECHO is reset.

CLS To Clear the Screen

One goal of batch file creators is to keep a clean screen. Some people prefer not to see even the first ECHO OFF when a batch file starts. Unfortunately, because batch files start with an implicit ECHO ON command, you cannot stop the first ECHO OFF statement from being displayed. However, you can use a trick that almost eliminates the display. You can start the batch file like this:

```
ECHO OFF
CLS
```

CLS is the DOS clear-screen command built into COMMAND.COM. Although not a batch subcommand, CLS has an excellent use in batch files. When CLS immediately follows ECHO OFF, the screen clears so quickly that you see the ECHO OFF only if you look for it. If you put these two statements at the beginning of the example, the screen looks like figure 7.1 as the batch file is executed.

```
To run this program, make sure the disk containing
BASICA is in drive A and the disk labeled
CONTRACTS is in drive B.
.............................................
(If you need to exit at this time, hold down the
Ctrl key and press the Break key. When the system
asks you whether you wish to "Terminate batch job (Y/N)?"
press Y. The batch file will stop.

C:\ -->
```

Fig. 7.1. *The effect of using ECHO OFF and CLS at the beginning of a batch file.*

Waiting for Action: PAUSE

As written, the preceding batch file just displays messages. To make the batch file wait for the user to take an action, you use the PAUSE subcommand. PAUSE stops the batch file, displays the message `Strike a key when ready...`, and waits for the user to do just that. When a key is pressed, the batch file marches back into action, picking up right after the PAUSE command. PAUSE also allows an optional message on the same line, for example,

```
PAUSE Insert the appropriate disk
```

This message is displayed only if ECHO is on. If you usually turn off ECHO, put needed messages in an ECHO command just before the PAUSE. However, the `Strike a key when ready...` message is displayed whether ECHO is on or off. You have no control over the display of this message.

Because this line always shows on the screen, you should attempt to phrase your batch file message to take advantage of this line. For example, you can change these lines:

```
ECHO OFF
CLS
ECHO Place the proper diskette into drive A and
ECHO when the message appears press Enter to continue
ECHO or press Ctrl-Break then Y to stop the file
PAUSE
```

You can rephrase your message like the following:

```
ECHO OFF
CLS
ECHO Place the proper diskette into drive A.
ECHO If the diskette is not available,
ECHO press Ctrl-Break then Y to stop the file or
PAUSE
```

The first set of lines yields the following screen:

```
Place the proper diskette into drive A and
when the message appears press Enter to continue
or press Ctrl-Break then Y to stop the file
Strike a key when ready...
```

The second set of lines displays

```
Place the proper diskette into drive A.
If the diskette is not available,
press Ctrl-Break then Y to stop the file or
Strike a key when ready...
```

Batch File Parameters

One useful technique is to write a general batch file that can be used under different circumstances. Most DOS commands are general-purpose commands. For example, DIR produces a directory listing of any disk drive or subdirectory. You choose which disk drive or subdirectory you wish to see by adding the drive or subdirectory name after the command. You simply separate the command from the disk drive or subdirectory name with a space (or a comma, colon, semicolon, single quotation mark, or equal sign). By accepting additional information on the line with the command, the command has greater usefulness. The same usefulness can be built into the batch files you create.

Defining Parameters

The information you type after the command (for example, **\DOCS** in the command **DIR \DOCS**) is called a *parameter*. Within a batch file, you can define up to 10 parameters. (More than 10 parameters can be used, as is discussed in the section about the SHIFT subcommand.) A parameter is defined within the batch file by using a *variable*, or *marker*, which is the percent sign (%) followed by a number from 0 to 9.

Look at the following example of the batch file, MOVE.BAT:

```
COPY C:\%1 C:\SUBDIR1 /V
ERASE C:\%1
```

Suppose that you type the command line

```
MOVE FORMAT.COM
```

The batch file MOVE.BAT begins executing and substitutes the first parameter (FORMAT.COM) for the variable %1. After the substitution, the lines in the batch file become

```
COPY C:\FORMAT.COM C:\SUBDIR1 /V
ERASE C:\FORMAT.COM
```

DOS changes each %1 to the first parameter from the command line.

If FORMAT.COM is located in the C drive root directory, the batch file copies FORMAT.COM (with verification) to the subdirectory called SUBDIR1 and then erases FORMAT.COM from the root. You can see why this batch file is called MOVE. The batch file copies a file from the root directory to a subdirectory called SUBDIR1 and then erases the file. This batch file saves you the extra step of erasing the original file after it has been copied.

You might be tempted to make this batch file more general by setting up two parameters: one for the file name and the second for the disk or directory to which the file will move. By using the second parameter, you can direct DOS to copy the file to any subdirectory on drive C. This new batch file is simple:

```
COPY C:\%1 C:\%2 /V
ERASE C:\%1
```

Then, to move the file FORMAT.COM to the directory C:\DOS31, you type

```
MOVE FORMAT.COM DOS31
```

But there is a major flaw lurking in this batch file. What if you mistype or omit the directory name, the second parameter? DOS "burps" on the first step, the copying of the file, and issues an `Invalid directory` error message. The file is not copied. The danger is that DOS happily proceeds to the next step of *erasing* the file. The original file has not been copied but is erased! In the same way if you forget to type any second parameter, DOS declares, `File cannot be copied onto itself` and does not copy the file. As before, though, the second line causes DOS to erase the file. The earlier MOVE.BAT file did not encounter this problem because a valid subdirectory name for the destination (SUBDIR1) was "hard-wired" (embedded) in the batch file.

This second batch file is less than forgiving about errors. As the new MOVE batch file stands, the generality comes with a steep price for mistakes. The more general MOVE batch file is not worth the potential problems and should not be used.

Counting Parameters

A batch file has 10 possible parameters: %0 through %9. The numbering scheme used for these parameters may be confusing for newcomers.

To understand the numbering, remember that computers use zero as a real counting number, not just a place holder. The first parameter is actually parameter number 0 and designated as %0 in batch files. The %0 is predefined for you—it is the name of the batch file as you typed it. For either of the command lines given, %0 is MOVE. The remaining items on the line are parameters 1 through 9. The first word after the batch file name is parameter number 1, %1 in the file. The second word is %2, and the ninth item on the line is %9. Each word on the line is separated from the next by a space, comma, colon, semicolon, single quotation mark, or equal sign.

Although DOS restricts you to parameters 0 to 9 within a batch file, a command line can have many parameters within the command line's 127-character limit. In the section about the SHIFT command you learn how to trick DOS into using the other arguments.

The other popular and interchangeable term for *parameters* is *arguments*. The computer definition of arguments is derived from the mathematical definition: independent variables on which a function's value depends. Technically, arguments for a batch file are parameters, but the distinction is irrelevant.

Looping within a Batch File: GOTO

The remaining batch subcommands are more analogous to programming commands and allow more sophistication in your batch files. The first "programming" batch command, GOTO, uses simple branching (jumping) to designated lines within the file.

Ordinarily, DOS runs a batch file one line after the other. With GOTO you can change the order of execution (the flow) of the lines in a batch file. The DOS GOTO command is like the BASIC programming language's GOTO. With GOTO, the flow of the batch file can go almost anywhere in the file.

To have some place to GOTO, you use a label. In a batch file, a *label* is a name that is preceded by a colon in the first column of the line. The following is an example of a "perpetual" batch file using GOTO and a label:

```
:START
DIR B:
PAUSE
GOTO START
```

As the first character in the line, the colon designates that the name START is a label. When this batch file is run, DOS displays a

list of the files on drive B and pauses. The final line directs DOS to go to the :START line and execute the batch file again.

This silly but illustrative batch file continues until you stop the action by pressing the Ctrl-Break or Ctrl-C key combination. (It isn't elegant to stop a batch file execution by using Ctrl-Break unless absolutely necessary; as I dive further into batch files, I'll show some cleaner methods of exiting.)

Labels can have as many characters as you like, but only the first eight characters have any significance to DOS. Although labels can be any length, stick to label names with eight or fewer characters. Additional characters in label names rarely help document what the batch file does.

Imposing Conditions: IF

Often in a batch file, you need to take action only under certain conditions. A single command or set of commands is executed or skipped based on a particular condition. For example, your batch file may need to react differently to different values of the command line parameters. For this and other situations, the IF subcommand provides the means to take action based on the result of a condition. The general form of the IF subcommand is

IF *NOT* **condition command**

When the optional *NOT* is specified, the command is executed if the *condition* is *not* true. You can test for three conditions:

1. The ERRORLEVEL of a program
2. The match of one string of characters with another string
3. The existence of a file

The *command* is any valid DOS command or batch subcommand except another IF statement. You cannot put two or more IF subcommands on the same line.

Using IF, you can modify the MOVE.BAT file so that it first checks to see if the file you want to move exists in the root directory:

```
IF NOT EXIST C:\%1 GOTO EXIT
COPY C:\%1 C:\SUBDIR1 /V
ERASE C:\%1
:EXIT
```

If the specified file (%1) does not exist in the root directory, the batch file jumps (the GOTO command) to the EXIT label and ends.

If the file does exist, the file executes the COPY and MOVE commands.

Because IF controls a single command, a common technique is to test for the opposite condition and jump around the lines you do not wish to execute. My preference would be to say that the file exists and then to copy and erase the file, but DOS does not have the ability to control two or more commands with IF. The alternative is to test for the opposite: "Does the file not exist?" If the answer is true, the file skips the COPY and ERASE commands. If the answer is false (the file exists), execution drops through the IF statement to the following lines and executes the COPY and MOVE commands.

Note that execution always reaches the :EXIT label, regardless of the IF test. No harm is done because DOS views labels benignly. When a label is the next line to execute, DOS just ignores the line and moves to the following line. Labels have meaning only as targets of GOTO statements; otherwise, DOS treats labels as harmless "noise."

If you want to get fancier, you can extend MOVE.BAT again by having the batch file display a "file does not exist" message, as follows:

```
ECHO OFF
CLS
IF NOT EXIST C:\%1 GOTO ERROR1
COPY C:\%1 C:\SUBDIR1 /V
ERASE C:\%1
GOTO EXIT
:ERROR1
ECHO The file (%1) does not exist . . .
:EXIT
```

This batch file shows the convoluted logic you must use because IF controls a single command. The third line performs the test for a file's existence. If the condition is true (the file does not exist), processing jumps to ERROR1, where the message states that the file does not exist. Then processing skips :EXIT and ends.

Notice that after the COPY and ERASE commands, execution also jumps to EXIT. The reason is simple. If the file did not jump to EXIT, DOS would display the "file does not exist" message. Displaying an error message when no error has occurred is not just poor technique. The display is disruptive to the batch file's user. Justifiably, the user may want to use a hammer at the next "strike any key to continue" message.

You can modify the batch file again to get rid of the problem in an earlier version of MOVE: "goofing" the COPY and then erasing the old file. One more IF command, an ECHO, and another GOTO handle the problem.

```
ECHO OFF
CLS
IF NOT EXIST C:\%1 GOTO ERROR1
COPY C:\%1 C:\%2 /V
IF NOT EXIST C:\%2\%1 GOTO ERROR2
ERASE C:\%1
GOTO EXIT
:ERROR1
ECHO The file (%1) does not exist . . .
GOTO EXIT
:ERROR2
ECHO The copy of C:\%1 to C:\%2 was unsuccessful!
ECHO C:\%1 was not erased.
:EXIT
```

This version tests whether the new file exists and is successfully copied. You combine C:\ with %1, the file name, and %2, the subdirectory that will hold the copied file. The result is a complete file name (C:\%2\%1), which is used in the test for existence. If the file exists, the COPY command was probably successful. If, however, anything is wrong, the batch file does not erase the original. This line is the safety play, using the approach of doing nothing destructive in case something might have gone wrong. When creating batch files, use the approach that everything must be right before destroying any files.

Checking Strings with IF

This section presents a practical example that illustrates the string-comparison option of IF. I created a special batch file called FORMAT.BAT after one of my associates accidentally reformatted his hard disk. Wanting to format a floppy diskette but not being familiar with DOS formalities, at the C prompt, he simply typed

```
FORMAT
```

Of course, because he did not specify the disk drive to format, DOS pre-V3.2 used the default disk, in this case C:, the hard disk drive. For some reason, my cohort panicked and ignored the warning the

DOS FORMAT command issues whenever it is about to format (or reformat) the hard disk.

I decided to take some simple action to ensure that an accident like this would not happen easily again. First, to hide, or disguise, the FORMAT.COM program, I renamed it XFORMAT.COM, a name of my own choosing, with the command

RENAME FORMAT.COM XFORMAT.COM

Now, if someone blithely types FORMAT, nothing drastic happens because no such program file exists on the hard disk. Then the following batch program was created and called FORMAT.BAT:

```
ECHO OFF
IF %1. == . GOTO :NONE
IF %1 == C: GOTO :NOCAN
IF %1 == c: GOTO :NOCAN
XFORMAT %1
GOTO END
:NONE
ECHO You did not specify the drive (B:), e.g. FORMAT B:
ECHO Please try the command again.
:NOCAN
ECHO You don't really mean to do that—format the
ECHO C drive—do you?
:END
```

I use the IF subcommand to test repeatedly for the equivalence of two strings. Remember that *strings* are defined as sets of arbitrary (any) characters of arbitrary (any) length. For example, each command line parameter is a string. When comparing two strings, DOS simply compares two sets of characters.

With the renamed file and batch file in place, typing **FORMAT b:** causes DOS to use the batch file, not FORMAT.COM, now named XFORMAT.COM. DOS substitutes *b:* for *%1*. The three IF lines become

```
IF b:. == . GOTO :NONE
IF b: == C: GOTO :NOCAN
IF b: == c: GOTO :NOCAN
```

Each IF test executes, finding that *b:* is not the same as ., *C:*, or *c:*. All three tests fail, and DOS drops to the command to execute FORMAT under its new name with the given drive name.

If, for any reason, someone explicitly attempts to format the C drive, line 3 traps the command **FORMAT C:**. Processing jumps to the label NOCAN, gives the hand-slapping message, and does not format the disk.

Notice the fourth line. DOS is literal in comparing strings. Capital *C* is not the same as lowercase *c*. Line 4 handles the difference by repeating the test with an uppercase C. Remember that in matching strings DOS does see a difference between the cases of characters. If you wish to test for both uppercase and lowercase letters, you must test for both cases.

Line 2 of this batch file handles the core problem that my friend created. His havoc-wreaking FORMAT command did not use any parameters. The line is less complex than you might expect. You might think that either of the following lines would also work.

IF %1 == GOTO NOCAN

IF %1 == '' GOTO NOCAN

You may think that the missing second string or the '' as the second string would designate an empty string, a string with no characters in it. However, DOS substitutes exactly what it "sees" for each parameter. In the case of giving no additional parameters to the batch file, the respective results are

IF == GOTO NOCAN

IF == '' GOTO NOCAN

In other words, DOS substitutes *nothing* for something. DOS is ingracious about comparing nothing to anything and gives a very explanatory `Syntax error` and pops out of the batch file.

To solve the problem, DOS must see something. A period was used in the batch file. When executing line 2 with no parameters, the line becomes

```
IF . == . GOTO NONE
```

The condition is true, and the program jumps to the message about needing a disk drive name. You could also use single quotation marks in this line:

IF '%1' == '' GOTO NONE

The line would become

```
IF '' == '' GOTO NONE
```

Either technique is usable and acceptable.

Checking ERRORLEVEL

The last condition the IF subcommand tests is the so-called ERRORLEVEL of a program. ERRORLEVEL is a numeric code between 0 and 255. The code is set by a program when the program exits. The code is passed to DOS and can be tested by other programs or by the IF subcommand. Many computerists call the ERRORLEVEL an *exit code*, a more appropriate name.

By convention, a program that does not encounter an error leaves an exit code of zero. If an error is encountered, the exit code is set to one or higher.

The general form to test the exit code of a program is

IF ERRORLEVEL n command

The value of *n* is a number, and *command* is the resulting action to be taken if the condition is true. ERRORLEVEL is different from the other two conditions for IF. The IF ERRORLEVEL statement is true if the exit code of the previously executed program is *greater than or equal to n*. In other words, this line

IF ERRORLEVEL 1 GOTO END

executes if the exit code of the program is 1 or greater. If the exit code is zero, the command does not execute.

To use this option, you must know whether the specific programs use exit codes and what the codes mean. For example, the only DOS-provided programs to use exit codes are BACKUP, RESTORE, and XCOPY. Other provided DOS programs do not leave exit codes. Many other programs, however, do leave exit codes. The following chapter, which deals with advanced batch files, provides several demonstrations of IF ERRORLEVEL.

Looping: FOR..IN..DO

The FOR..IN..DO subcommand adds the power of repetitive looping to the batch subcommands. In other words, you can have the batch file DO something FOR every value of a variable IN a specified set of commands. The general form of the subcommand is

FOR %%variable IN (set) DO command

The *variable* is a single-letter name, which the programmer enters. Any letter can be used. I prefer to use lowercase letters so that the lines are more easily read. The double percent sign (%%) is

important. The *%%* distinguishes this variable from a batch parameter, which uses a single percent sign. The *set* enclosed in parentheses is an item or name, or a list of items or names, that you want to use in the FOR statement. Several "names" (words or file names) can be used in *set* providing the names are separated by spaces or commas. Wild cards (* or ?) in file names are also valid here.

A simple example of FOR..IN..DO gets a directory listing for each file in the directory. The batch file is

FOR %%a IN (*.*) DO DIR %%a

For every file that matches the name *.* (all files), DOS does a DIR command (directory list) using the names of the matching files. If the directory holds COMMAND.COM, AUTOEXEC.BAT, and CONFIG.SYS, DOS expands the batch file line in the following manner:

FOR %%a IN (COMMAND.COM AUTOEXEC.BAT CONFIG.SYS) DO DIR %%a
DIR COMMAND.COM
DIR AUTOEXEC.BAT
DIR CONFIG.SYS

DOS executes the DIR command once for each file in the set. Each time the command is executed, DOS substitutes for the %%a an item from the set. In this case, the DIR command is executed three times sequentially using the names in the set. In sequence, %%a is replaced by COMMAND.COM, AUTOEXEC.BAT, and CONFIG.SYS.

This batch file is simplistic and impractical because a single DIR command shows a listing for all files in the directory. However, the file is an example of how FOR..IN..DO works.

You are not restricted to file names. You can use other names, such as disk drive names, path names, or just plain words. Based on this principle, lines 3 and 4 of the FORMAT.BAT file can be replaced with a single line:

FOR %%d IN (C:,c:) DO IF %1 == %%d GOTO NOCAN

The FOR..IN..DO command executes once for each member in the *set*. DOS executes the line in this manner:

```
FOR %d IN (C:,c:) DO IF %1 == C: GOTO NOCAN
IF %1 == C: GOTO NOCAN
IF %1 == c: GOTO NOCAN
```

As you can see, *%%d* (or any other letter) is a variable name for which the command substitutes the values in the parentheses, C: and

c:. The following example also allows you to switch the flow of the batch file by testing for valid disk drive names rather than invalid names. The new FORMAT file is

```
ECHO OFF
IF %1. == . GOTO NONE
FOR %%a IN (a:, A:, b:, B:) DO IF %1 == %%a GOTO FORMAT
ECHO You don't really mean to format %1, do you?
GOTO END
:NONE
ECHO You did not specify the drive (B:), e.g. FORMAT B:
ECHO Please try the command again.
GOTO END
:FORMAT
XFORMAT %1
:END
```

The flow of this batch file is slightly different. You must first test for an empty parameter *%1* because DOS gives errors when testing anything against an empty string. The next command tests whether the designated disk drive is A or B, in both upper- and lowercase versions. If the letters match, processing jumps to where the file invokes the FORMAT command. If no match is found, processing drops to an error message then jumps to END.

This batch file is slightly preferable to the former FORMAT.BAT because this version allows only drives A and B to be formatted. The other batch file prevents drive C from being formatted but not drives D and up, if such disk drives exist.

Getting More than 10 Parameters: SHIFT

As stated earlier in this chapter, DOS allows the explicit use of only 10 batch parameters, %0 through %9. However, DOS can be tricked into using more parameters by use of the batch subcommand SHIFT. Placing SHIFT in a batch file moves each command line parameter one position to the left. In other words, parameter 0 (zero) is discarded, old parameter 1 becomes new parameter 0, old parameter 2 becomes new parameter 1, and so on.

SHIFT is commonly used when you are handling an undetermined number of similar parameters. For instance, I could take the earlier MOVE.BAT file, the one that moved files to a predetermined subdirectory, and generalize the MOVE.BAT file so that it could move any number of files.

```
ECHO OFF
CLS
:START
IF NOT EXIST C:\%1 GOTO ERROR1
COPY C:\%1 C:\SUBDIR1 /V
ERASE C:\%1
SHIFT
IF %1. == . GOTO EXIT
GOTO START
GOTO EXIT
:ERROR1
ECHO The file (%1) does not exist . . .
:EXIT
```

To move any number of files, you can enter a command like

MOVE COMP.COM ABACUS.EXE SLIDE.EXE LINK.EXE ABSCOND.COM

MOVE.BAT substitutes the first file (COMP.COM) for *%1*, processes the file, and shifts the parameters so that the second file (ABACUS.EXE) now becomes *%1*, and so on.

The most frequent use of SHIFT is to do the same work on an unknown number of files, for example, running the Macro Assembler on any set of files. The batch file that does this work looks similar to the MOVE file.

```
ECHO OFF
CLS
:START
IF %1. == . GOTO EXIT
IF NOT EXIST %1 GOTO MISSING
MASM %1;
SHIFT
GOTO START
:MISSING
ECHO %1 does not exist . . .
SHIFT
GOTO START
:EXIT
```

As in the previous batch files, ECHO is turned off and the screen cleared. DOS immediately tests for an empty parameter and jumps to the EXIT if *%1* is empty. Processing goes to a warning message if a named file does not exist. Otherwise, DOS invokes the Macro Assembler, MASM.EXE, with the correct arguments. The file then

shifts the batch file parameters one position to the left and loops back to the test for an empty parameter.

Notice in this batch file that if a file name is wrong, we inform the user about the error but continue to work on the other files. Nothing is more infuriating than having to retype a complete command line when just one file name is wrong. Because MASM does not alter the original files, the "keep on trucking" approach for this application is safe and practical. The batch file creator should temper the operational philosophy of the batch file according to what the batch file must do and who will use the batch file.

Summary

This chapter has covered the basics of batch files. The important points to remember are

- Batch files hold ASCII text and have an extension of BAT.
- You execute a batch file by typing its root name.
- The batch file can contain any valid DOS command and the seven batch subcommands.
- The REM subcommand allows you to place messages or remarks in a batch file.
- The ECHO subcommand has two uses. ECHO turns on or off the display of messages for the batch file. When ECHO is OFF, the display of commands and REMs and other batch file messages is inhibited. ECHO ON enables the display. The ECHO message unconditionally displays a message.
- The DOS CLS command can be used to clear the video display, at any DOS command line or from within a batch file.
- The PAUSE subcommand is used to wait for a keystroke from a user. PAUSE can display an optional message, which can be suppressed by an ECHO OFF statement. The PAUSE message `Strike a key when ready...` is always displayed.
- The GOTO subcommand moves the flow of execution within the batch file to a label, a name starting with a colon. DOS views only the first eight characters in a label name.
- The IF statement performs conditional testing. You can test for a file's existence, the match of two strings, or the ERRORLEVEL (exit code) of a program. You can examine for the opposite condition (not existing, not equal, less than) by using the

optional NOT key word. The IF subcommand controls a single command.

- The FOR..IN..DO subcommand allows you to repeat a command once for each file name or word in a given set. Like IF, FOR..IN..DO controls a single command.
- With the SHIFT subcommand, you can have more than 10 parameters per line.

In the next chapter, you will extend your knowledge of batch files with more examples and more complex techniques, including a simple menu system.

Chapter 8

Advanced Batch File Techniques

This chapter explores some advanced techniques using batch files. You will find tips on avoiding some common mistakes when using batch files, and you will learn some new ways to use the start-up batch file, AUTOEXEC.BAT. You also will learn some tricks for running batch files—by using COMMAND and by running batch files from within other batch files. The chapter also explores some nonDOS programs and contains several skeletal batch files for a menu system.

Avoiding Common Pratfalls

It is easy to make several humbling errors when using batch files. This section alerts you to some of these errors and gives suggestions for avoiding them.

Using ASCII Text To Create Batch Files

Remember that batch files use ASCII text; if you try to create a batch file with nonASCII text, you will have trouble. You should create the batch files with the COPY command (as described in Chapter 4), the DOS line editor EDLIN, some other text editor, or a

word processor using the nondocument or programmer's mode. Some word processors produce documents with scrambled text. DOS does not react kindly to this type of file.

Some word processors maintain files in a scrambled ASCII mode but print ASCII text. Most word processors have an option to print text to a disk-based file. If your word processor lacks the nondocument or programmer's mode, create a file and then print the text to a disk file. Make sure that any internal formatting codes the word processor uses are turned off. Check the resulting "printed" file using DOS's EDLIN and TYPE commands. If the text is readable and does not contain strange characters, you can use the print-to-disk technique to create batch files.

Using Absolute Drive and Path Names

If possible, use absolute disk drive and path names in batch files. By using the PATH command or prefixing a disk drive/path name to a command or batch file name, you can run a batch file that "is anywhere from anywhere." When running the batch file from a different directory or disk drive, DOS can fail miserably if your file names use relative paths or don't use disk drive names. Look at this simple example:

```
CD \SPECIAL
SPECPROG
CD \
```

This batch file works perfectly on one assumption: the current disk drive has a subdirectory called \SPECIAL. Because the current disk is normally the hard disk, the assumption is usually correct. But if the current disk drive is a floppy drive, the batch file fails. A better batch file is

```
C:
CD \SPECIAL
SPECPROG
CD \
```

This batch file ensures that C is the current disk drive.

On occasion, you will want the hard disk's current directory to be the directory holding your program, but you will want the current disk to be the floppy disk. For instance, you may want to quickly edit a file on a floppy disk without copying the file to the hard disk. Rather than invoke the program and then change the current disk drive to the floppy disk drive, you first make the floppy disk the

current disk drive. Then you use the following batch file to invoke the word processor:

```
CD C:\SPECIAL
C:SPECPROG
CD C:\
```

Notice the explicit disk drive name added to each line. In this batch file, you do not change disk drives; you simply ensure that each command affects drive C. You use this batch file to invoke the applications program without changing the current disk drive.

For similar reasons, it is wise also to use absolute path names in your batch files. When the batch file is run from different directories, you must use absolute path names in the file. If you do not, DOS may look for subdirectories that do not exist in your current directory. The price of this mistake is not harmful failure, just failure—the batch file doesn't work.

Premade Batch Files

Always examine "premade" batch files that come with commercial programs; the file may not do exactly what you want it to do. Use the TYPE or MORE command to look at the file for actions you may not want or actions you need to add.

Typical premade batch files that come with commercial programs are those used for installing or invoking the program. Many of these batch files are flexible; they allow you to install programs anywhere on your hard disk and to use the programs from anywhere. The batch file, however, may have some operational aspects you want to change. Your choices are to edit the old batch file or to add a "front-end" batch file to do the needed work before the manufacturer's batch file runs. Either choice is acceptable.

If you need to edit the commercial batch file, create a new batch file with a slightly different name, edit the contents to include or exclude the required command(s), then routinely invoke the altered batch file. This method allows you to fall back on the original batch file when needed.

The other choice is to "front-end" the original batch file. This technique requires an understanding of how to use batch files to run other batch files, as explained in the following section.

Indirect Ways To Run Batch Files

The most direct method of running a batch file is to invoke the file at the system prompt. But you can save time and effort, and have some fun with your computer's capabilities, by using indirect methods to run batch files. This section discusses running batch files from other batch files and by using COMMAND.

Running Batch Files from Other Batch Files

One batch file can run another batch file, but the direction of control is one way only. Batch files have no GOSUB command. When the first batch file invokes the second, DOS does not allow the second batch file to return to the preceding batch file. The only way to loop back to the first batch file is to run the first batch file explicitly from the second.

Because of this limitation, the usual technique is to do as much as possible in the first batch file and use the last step of the first batch file to call the second batch file. For example, suppose the manufacturer provides the batch file RPS.BAT to start the program; you need to go to the directory holding RPS before you run that file. You could construct a batch file, say SPR, with these contents:

```
C:
CD \RPS
RPS
```

This batch file changes the current disk drive and current directory to C:\RPS and runs the RPS batch file.

If you want the RPS batch file to rerun the first file, you add the following line to the end of the RPS file:

```
SPR
```

The second batch file automatically invokes the first batch file. The batch files will continue to loop from one to the other until you press Ctrl-Break or Ctrl-C.

The technique lacks eloquence, but it works in simple situations. For more complex situations, this technique is inadequate. For example, it does not address the occasions when you must run other batch files from the middle of one batch file and then return to the middle of the first. DOS does not have a GOSUB command to solve

this problem for batch files, but you can approximate the GOSUB by using some tricks with COMMAND and COMMAND.COM.

Using COMMAND To Run Batch Files

You can use DOS's command processor in conjunction with COMMAND to simulate the GOSUB command. The technique permits you to run batch files from any position within other batch files. To grasp the technique, you should first understand how COMMAND.COM functions within DOS.

COMMAND.COM is the DOS command processor. You can invoke a different command processor if you want to, by using the SHELL command (see Chapter 4 for details). Although COMMAND.COM is the third major memory-resident part of DOS, COMMAND is not sacred to DOS. COMMAND is a program, which you can run for your uses. The syntax for COMMAND is

COMMAND */P /C string*

/P is the *permanent* switch. This switch is identical to SHELL's */P* switch. When you issue COMMAND with the */P* switch, an additional copy of the command processor is created and stored in memory until you restart DOS. If you don't give any switches to COMMAND, COMMAND acts as if the */P* switch is on, and the additional copy of the command processor stays in memory. If you have not used the */P* switch, you can force DOS to discard the additional copy by giving the EXIT command.

The */C* switch tells COMMAND that a *command* is to be executed. COMMAND.COM executes whatever is in *string*, usually a command to invoke a program or a built-in DOS command. After the command is executed, COMMAND.COM automatically exits.

The */P* and */C* switches have one restriction: you cannot use them together.

To test using COMMAND for your own purposes, try the following procedure, taking the steps in order. Keeping in mind that additional copies of COMMAND.COM take up some free memory, use the CHKDSK (check disk) command to get reports on free memory, and use COMMAND to generate copies of COMMAND.COM. The decrease in free memory reported by CHKDSK indicates that a copy or copies of COMMAND.COM are "permanently" in memory. When running CHKDSK the first time, jot down the amount of free memory CHKDSK reports; then compare this amount when running CHKDSK at other times.

```
CHKDSK
COMMAND /C CHKDSK
CHKDSK
COMMAND
CHKDSK
EXIT
CHKDSK
```

The second and fourth times that CHKDSK runs, the amount of free memory is lower. The third and last times CHKDSK is used, however, it shows the same amount of free memory as the first run of CHKDSK. The third run of CHKDSK proves that when you give the */C* switch with COMMAND, another copy of COMMAND.COM loads into memory and executes the given command; the additional copy then "disappears." The difference between the fourth and fifth runs shows that if no switch is given, the additional COMMAND.COM stays in memory until you give the EXIT command.

This process is called *running another shell* because the UNIX community calls its command processor the SHELL. The SHELL is the origin of DOS's CONFIG.SYS and the BASIC programming language commands. The heart of the process involves invoking another copy of COMMAND.COM to run your command. Then the second copy exits, returning control to the original COMMAND.COM. Using other terminology, you *exec* another copy of the shell (COMMAND.COM) so that it can execute another program. The DOS world has inherited this terminology from UNIX also; *to exec* means to execute a program.

Using COMMAND To Run Other Batch Files

Now that you understand how to run another shell, you can begin to use COMMAND as a batch file GOSUB command. In later sections in this chapter, you learn to construct a menu system that uses the COMMAND trick.

Suppose that you have a batch file called MAIN.BAT. If you specify *S* for the first parameter when you run MAIN.BAT, DOS executes a second batch file called SUB1.BAT; otherwise the second file is not executed. Let us say, however, that you need to execute additional commands whether or not you execute SUB1.BAT. Here's the skeleton of MAIN.BAT:

```
ECHO OFF
CLS
    .
    .
FOR %%d IN (s,S) DO IF %%d == %1 COMMAND /C SUB1
    .
    .
```

If %1 equals *s* or *S*, you invoke a second copy of COMMAND.COM to run SUB1.BAT. When the second copy of COMMAND.COM finishes with SUB1.BAT, the second copy exits and the first copy of COMMAND.COM continues execution of MAIN.BAT. In effect, COMMAND.COM allows a GOSUB to a second batch file.

You can change the example slightly. Suppose that MAIN.BAT should always run SUB1.BAT. After SUB1.BAT is finished, SUB1.BAT should automatically restart MAIN.BAT. Instead of adding the line MAIN to the end of SUB1.BAT, you can use the GOTO statement to create an endless loop. Rather than change SUB1.BAT, you change MAIN.BAT like this:

```
:START
ECHO OFF
CLS
    .
    .
COMMAND /C SUB1
GOTO START
```

In this example, MAIN.BAT uses COMMAND to run SUB1.BAT. When SUB1.BAT finishes, the second copy of COMMAND disappears and the first copy of COMMAND continues execution of the lines in MAIN.BAT. The GOTO START line is executed and DOS jumps back to START and executes the line in MAIN again. This loop continues until the user presses Ctrl-Break or Ctrl-C to stop the batch file.

You need to remember three points when running COMMAND.COM. First, COMMAND.COM must be in the current directory or in a directory specified by the PATH command. If this is not the case, you must add the needed drive and path names to the beginning of the COMMAND /C SUB1 line.

Second, each new COMMAND.COM starts with ECHO ON. If you run additional batch files, you must set ECHO OFF at the start of each batch file that COMMAND.COM runs.

Finally, each layer of COMMAND.COM temporarily uses some RAM. Based on the version of DOS and your CONFIG.SYS setup, each

copy takes about 18K (DOS V2) to 26K (DOS V3). When the additional copy of COMMAND.COM exits, the RAM is freed.

Suppose you alter MAIN to run several batch files, like this:

```
.
.
COMMAND /C SUB1
COMMAND /C SUB2
COMMAND /C SUB3
```

Only one additional copy of COMMAND.COM is active at any time. Therefore, you temporarily lose about 20K of memory while SUB1, SUB2, or SUB3 is running. If, on the other hand, SUB1, SUB2, or SUB3 were to invoke additional copies of COMMAND, or if these files were to run a program needing every byte of RAM possible, you would temporarily trap layers of COMMAND.COM. In doing so, you also trap temporarily some RAM.

Most people use computers with 512K or more of memory, so having two or three copies of COMMAND.COM in memory is seldom a problem. Trapped COMMAND.COMs, however, might deny some spreadsheet or database users the amount of available RAM they need.

Although using COMMAND as a GOSUB command consumes some RAM, this technique is particularly useful when you are constructing menu systems. Later in this chapter, you will see a menu system that uses this technique.

The AUTOEXEC Batch File

As you learned in Chapter 4, DOS automatically executes the AUTOEXEC.BAT batch file whenever DOS starts from a power-up or system reset (Ctrl-Alt-Del). The AUTOEXEC.BAT file set up in Chapter 4 was functionally spartan. The ideas in this section dress up your AUTOEXEC file and make the computer more useful or just plain fun to use.

Using a New PROMPT

By now, you are familiar with the standard DOS prompts: C> for a hard disk system, A> or B> when the current disk drive is a floppy. Within one week of using a computer with a hard disk, you probably will face the problem of keeping track of what subdirectory you are

in. The default system prompt tells you the current disk drive but gives no clue to the current directory. Endlessly typing CD to get the current directory is painful. Changing the system prompt is not.

The DOS PROMPT command is flexible and allows a wide range of information to be given by the system prompt. Do you want to use both the current disk drive and the current directory as your prompt? Here's the command you can use in your AUTOEXEC.BAT file:

PROMPT $p –$g

The *$p* is a code name that represents the current path including the disk drive name. The *$g* is the greater-than (>) sign. Put them together and you get a prompt like this:

```
C:\DOS31 -->
```

There is a minor disadvantage in using this prompt or other prompts that show the current directory. If the current disk drive is a floppy disk drive, DOS always checks the diskette to see whether the current directory is valid. DOS normally reacts kindly to changing floppies "on the fly," but floppy diskettes rarely have the same subdirectories. DOS checks the diskette to see whether the same subdirectory exists. This means DOS tries to read the floppy diskette each time before displaying the system prompt. If you don't have a diskette in the disk drive, PC DOS gives you the following message:

```
Drive not ready
Abort, Retry, Ignore?
```

Don't bother trying the Abort or Ignore options. PC DOS won't do either. Nor can you change back to drive C. PC DOS is not satisfied until you put a diskette into the floppy disk drive. Then you can press *R*, *A*, or *I*.

Many MS-DOS implementations handle this problem gracefully. The error message is displayed, but you are allowed to enter the *A* for Abort. MS-DOS then changes to the root directory of the nonexistent floppy diskette and pops to the system prompt. You then can switch current disk drives back to the hard disk.

The general form of PROMPT is

PROMPT *prompt-text*

If you don't give any prompt-text, DOS reverts to C> style for the system prompt. Prompt-text can be any information. This information can include code names (called *metastrings*), which begin with a

dollar sign and use one additional character (called a *metacharacter*). Table 8.1 shows the metacharacters and their results.

Table 8.1
Metacharacters

Metacharacter	*What you get*
$	$ (the dollar sign)
_ (underscore)	carriage return/line feed, goes to the first character of the next line
b	| (the vertical bar)
d	the current date
e	the Escape character
g	> (the greater-than sign)
h	the backspace character, (Control-H)
l	< (the less-than sign)
n	the letter of the current disk drive name
p	the current path (including disk drive name)
q	= (the equal sign)
t	the current time
v	the version of DOS you are using
anything else	a null (empty) character

The following prompt satisfies most users' desire to obtain a proper sense of humility from their computer:

PROMPT p_yes, Don? $

The prompt prints the current path, drops a line, and prints the remaining text in this fashion

```
C:\
yes, Don? _
```

The final dollar sign ensures that a space will appear between the question mark (?) and whatever is typed at the new system prompt.

You use the *$e* metastring to place the Escape character in the prompt. The Escape character triggers the many features of ANSI.SYS. The following prompt places the current date, time, and path on the top line of the screen, and then places the *yes, someone?* message on the current line.

PROMPT $e[s$e[H$d $t pe[uyes, Don? $

PROMPT is a harmless command that can be used to make the computer more mentally and emotionally "comfortable" to use. Experiment with PROMPT. You can always revert to the original system prompt by issuing the PROMPT command without a prompt string. You can add the PROMPT command that generates your favorite system prompt to your AUTOEXEC.BAT file. Whenever the system is powered-up, your prompt appears.

Starting Up Memory-Resident Programs

You probably know something about these programs: SideKick®, ProKey™, SuperKey™, SmartKey™, Ready!, PC-Outline, Turbo Lightning, and Norton Commander™. The programs are desktop organizers, keyboard enhancers, outliners, spelling checkers, or DOS helpers.

These programs have one thing in common: each is a *memory-resident* program. You run the program, and the program snuggles up against DOS and springs into action when you enter certain keystrokes. Technically, these are called *terminate and stay resident* programs. The programs stop running and return control of the computer to DOS. The programs do not stop running completely, however; they monitor the keyboard for their "curtain call." Another popular term attached to these programs is *backgrounders*. They allow a regular program to run in the foreground while they hide in the background. When activated, the program springs into the foreground, suspending the current program until the resident program relinquishes control.

The AUTOEXEC.BAT batch file is the ideal place to enter the commands to invoke background programs. If you use more than one such program, be sensitive to the order in which they are loaded. As of this writing, these programs have no standard ground rules, and a certain loading order must be maintained or the programs do not work together correctly.

Another trick you may need is to switch to the directory that holds the background program. For example, do not start SideKick and SuperKey from just any directory. You should start these two programs from the directory in which the programs are located. Neither program will correctly locate its help files nor allow you to permanently change its features if you violate this rule. If SuperKey and SideKick are located in a directory called C:\BORLAND (Borland International is the publisher of both programs), you enter the following sequence in AUTOEXEC.BAT:

```
.
.
CD C:\BORLAND
KEY
SK
.
.
```

When you are booting from the hard disk, all works well.

Firing Up RAM Disks and Print Spoolers

RAM disk software cordons off a section of your machine's available RAM and makes that section "look like" another disk drive to DOS. When the computer writes to or reads from this "disk," DOS performs high-speed memory-to-memory transfers, which are significantly faster than transfers between memory and the floppy or hard disk drives.

A print spooler is a program that fools the computer into treating a portion of RAM as the place the computer should send any output to be printed. The computer sends output to this RAM buffer area at high speed, often freeing the computer sooner for other purposes. The buffer area feeds the information to the printer.

You can place RAM disk or print spooler start-up programs into the AUTOEXEC.BAT file *providing* you want them installed every time you boot up. You face trade-offs, however. RAM disks and print spoolers use the same RAM your programs use. Devoting this much RAM to a RAM disk and print spooler may keep you from using the full capabilities of programs like spreadsheets. The use of a RAM disk and print spooler can be application-specific; that is, their use can depend on the other programs you are running.

This is the guideline you should follow: If you consistently use applications that depend on the memory your RAM disk or print spooler would occupy, do not run the RAM disk or print spooler from the AUTOEXEC.BAT file. If, on the other hand, you normally can run a RAM disk or print spooler without running short of memory, start the programs from the AUTOEXEC.BAT file.

VDISK, the RAM disk software provided with DOS V3 and later versions, is started through the DEVICE command of CONFIG.SYS. In this case, no entry is needed in the AUTOEXEC.BAT file. DOS installs VDISK in memory before running the AUTOEXEC.BAT file. VDISK is discussed in Chapter 12.

Modest Menuing

Once you have a variety of applications that you use, you can make it easier to run them. At the very least, you may want to include in your AUTOEXEC.BAT file a method to display a neat list of available applications and the command(s) needed to invoke the applications.

For instance, if you routinely use Lotus 1-2-3, WordStar, dBASE III, CHART-MASTER® and Microsoft Project, you can create a batch file called MENU.BAT with the following lines:

```
ECHO OFF
CLS
:START
 .
 .
ECHO   * * * * * * Welcome to the IBM AT * * * * * *
ECHO   ==========================================
ECHO   Here is the list of available applications --
ECHO     To run:                  Type:
ECHO     ---------                ----------
ECHO     WordStar                   WS
ECHO     1-2-3                      LOTUS
ECHO     dBASE III                  DB
ECHO     ChartMaster                CM
ECHO     Microsoft Project          PROJ
ECHO   * * * * * * * * * * * * * * * * * * * * * * * * *
```

WS, LOTUS, DB, CM, and PROJ are the names of batch files that you have set up to switch to the appropriate subdirectory, call up the correct program, and so on. This batch file enables you to see at a glance the choice of applications available and the means to start each application.

You need two additional steps. The last line of each batch file used to run the application should read MENU. MENU should also be the last line of your AUTOEXEC.BAT file. When each program exits, the "driving" batch file (the file that invokes the program) automatically restarts the menu batch file.

A disadvantage to this batch file is that DOS uses more time reading each line and executing each ECHO statement than it takes to just type a file name. An effective alternative is to use a text editor to place the menu information into a separate file, say MENU.SCR, and use DOS's TYPE command to display the information.

In this case, MENU.SCR would look something like this:

```
* * * * * * Welcome to the IBM AT * * * * * *
==========================================
Here is the list of available applications --
     To run          Type
     -----------     --------------
     WordStar            WS
     1-2-3               LOTUS
     dBASE III           DB
     ChartMaster         CM
     Microsoft Project   PROJ
* * * * * * * * * * * * * * * * * * * * * * *
```

The new MENU.BAT file would look like this:

```
ECHO OFF
CLS
:START
.
.
TYPE MENU.SCR
.
.
```

Because all ECHO message statements are replaced by the one TYPE command, DOS executes this file much faster.

Another alternative makes the menu selection process easier. Number each of the menu selections as in the following example:

```
* * * * * * Welcome to the IBM AT * * * * * *
==========================================
Here is the list of available applications --

   1 WordStar
   2 1-2-3
   3 dBASE III
   4 ChartMaster
   5 Microsoft Project

Type a number to run a program...............
* * * * * * * * * * * * * * * * * * * * * * *
```

To make this system work, create a series of batch files—1.BAT, 2.BAT, etc.—with each numbered batch file corresponding to the relevant menu choice. (File names need not be only alphabetic characters; numbers work just as well.) A user who sees the menu

and types a number invokes a batch file that runs the selected program. For the sake of completeness, you can put as the last line in each of those batch files

TYPE MENU.SCR

This causes the menu to be redisplayed after each application.

It is strongly recommend that you place the MENU.BAT file in a PATHed subdirectory, one that is specified by the PATH command. This placement ensures that the MENU batch file can be run from anywhere. MENU.SCR be placed in a fixed location, probably the same directory as MENU.BAT. Also, you should prefix any mention of MENU.SCR in a batch file with the full disk drive and path name. If MENU.SCR is in C:\BIN, you use the following command to type the file:

TYPE C:\BIN\MENU.SRC

This ounce of drive and path name prevents a pound of insecurity should a user ever be "stranded" (left by a program) in an unexpected subdirectory.

These are a sampling of tools for constructing some simple menus to help manage your programs on a hard disk. Chapter 13 discusses other hard disk menu and file-management programs, as well as some public-domain, shareware, and other commercially available products.

Setting Up Help Files

Simple menus are a useful feature, but you can set up batch files that provide fuller explanations or documentation of menu choices, DOS commands, and so on. These help files can be particularly useful if you are setting up a system for a novice computer user.

To establish this help system, you create a batch file to display the help messages, and you create a file holding the text for the help. All files—batch and help messages—should be created using "vanilla" ASCII text.

The names of each help file should be consistent with the way the corresponding programs are invoked. If you are using the menu system described previously in the first example, you can call the files 123.HLP or WORDSTAR.HLP. If you are using the menu system with numbers, the files should be called 1.HLP, 2.HLP, and so on. All help files, except the batch file, should be placed in one subdirectory, such as \HELP or \BIN\HELP. The extension .HLP is

commonly used by programs for their help messages. By isolating your help files in a specially created subdirectory, you eliminate the risk of overwriting the manufacturer's help files.

Now create the HELP.BAT file as follows:

```
ECHO OFF
CLS
IF EXIST C:\HELP\%1.HLP GOTO OKAY
ECHO SORRY, HELP DOES NOT EXIST FOR %1
GOTO END
:OKAY
TYPE C:\HELP\%1.HLP
:END
TYPE C:\BIN\MENU.SCR
```

To redisplay the menu after the help file is displayed, you add the line

```
TYPE C:\BIN\MENU.SCR
```

to the bottom of the batch file. To use the help file to obtain more information about a particular program, the user types

HELP programname

or

HELP number

For constant reinforcement, you can add lines to your MENU.SCR file such as

```
or type HELP abbreviation to see suggestions
on the desired program.
* * * * * * * * * * * * * * * * * * * * * * *
```

or

```
Type a number to run a program.............
or to get help on a specific program, type
HELP number
* * * * * * * * * * * * * * * * * * * * * * *
```

Some forethought is necessary if the name of any help subject is more than eight characters long. Because batch file root names can be only eight characters long, you need some abbreviation or mnemonic device that the user can easily remember and use. Also, be careful if there are several probable names for the subject, such as

1-2-3, 123, or even LOTUS. This second problem is easier to handle than long program names. You can duplicate the help file using several names. Then create the help file using one name and copy the file to the second or third name.

If the help file is long (more than one screenful), you use this line for typing the help files:

TYPE C:\HELP\%1.HLP | MORE

Chapter 6 introduces the vertical bar (|) as the pipe character, allowing the output from one program to become the input to a second program. The preceding line directs the output of the TYPE command to the input of the MORE.COM program.

The MORE program presents a screenful of information then asks

```
--More--
```

When a key is pressed, MORE presents the next screenful. The process continues until the last screenful of information is shown.

DOS's I/O redirection is used to pipe information from the TYPE command into MORE. MORE presents the information one screenful at a time, allowing the user to take a casual stroll through the help information. Should the help file have fewer than 23 lines (one screen's capacity), MORE simply displays the lines and ends without asking for more. The help file display is clean and useful.

If MORE.COM is not in the current directory or on the PATH, give DOS the proper location by including the disk drive and path name for MORE as follows:

TYPE C:\HELP\%1.HLP | C:\BIN\MORE

As this help system is for infrequent or novice users, we do some bulletproofing of the batch file. The IF line in the HELP.BAT file is the safety check that the HELP file exists before we try to TYPE the file out. If someone types *HELP* with no file name, DOS will not find a match with an existent file name. Either way, the batch file displays the line

```
SORRY, HELP DOES NOT EXIST FOR name
```

The name typed on the command line is substituted for **name**.

This bulletproofing has a distinct drawback. The message does not tell the user what are the valid names for invoking HELP. The displayed message is even less helpful if the user omits the file name:

```
SORRY, HELP DOES NOT EXIST FOR
```

Help for what? The batch file does not handle empty parameters congenially. A better technique lays a trap for a missing name and responds accordingly. An even better technique displays the potential help names the person could use. Based on these suggestions, improve the HELP.BAT file as follows:

```
ECHO OFF
CLS
IF %1. == . GOTO LIST
IF EXIST C:\HELP\%1.HLP GOTO OKAY
ECHO SORRY, HELP DOES NOT EXIST FOR %1
:LIST
ECHO HERE ARE THE AVAILABLE HELP FILES -
DIR C:\HELP\*.HLP /W
GOTO END
:OKAY
TYPE C:\HELP\%1.HLP | MORE
:END
```

The example uses the DIR command in its five-wide format (/W option) to display the names of the help files. This method has one drawback: DIR prints the root name with the file extension. DIR displays names like 123.HLP, but you must type *123* without the HLP extension to invoke the appropriate HELP. You may wish to rename all HELP files and change *.HLP to *. if you think seeing HLP on the screen will confuse your users. Second, eliminate all occurrences of HLP from the batch file, except for the DIR line. Change this line to either

```
DIR C:\HELP\*. /W
```

or

```
DIR C:\HELP\*. /P
```

The first line uses the five-wide display. This method still has a quirk—a period is displayed after the root file name. The second line displays the files in a vertical list and pauses if more than a screenful is displayed. No period appears after the root name, but the user gets more potentially confusing information (the date, time, file size, etc.).

The period after the asterisk is critical. Giving a single * to DIR is the same as giving *.*. This command is the only DOS command that matches the root and extension name when you use wild cards. If you give any extension to DOS, even an empty extension, DOS restricts the match. Therefore, *. matches only those files without any extension.

The other step to completing the help and menu task is ensuring that no files in the HELP directory are without extensions; otherwise, the user gets another opportunity for confusion.

Interactive Batch Files

To state the problem mildly—DOS batch files are only slightly interactive. The extent of our interaction with a batch file is to start the file, wait for the pause and press a key to continue, and press Ctrl-Break or Ctrl-C to halt the execution of the file. A gaping hole in DOS is the inability of a batch file to elicit input from the user and take action accordingly, as any program can do.

The first menu system we created suffers from this lack. You must exit from the batch file and then type the name of the batch file at the DOS system level. The batch file creator has little opportunity to catch errors. The user has every opportunity to create errors.

Another trick is needed to allow users to enter information the batch file can actively use. Fortunately, the magic is ready.

The public domain has several programs with the sole function of accepting characters or numbers from the user and leaving an exit code. By using one of these programs in conjunction with the IF ERRORLEVEL subcommand of DOS, you can construct interactive batch files. You insert the command to run the small program, supplying an appropriate prompt to the user. Then you examine the exit code using ERRORLEVEL to branch to the proper line in the batch file. Later in this chapter, you will be introduced to a public domain program that can accept names or words other than a name.

Before going further, you should understand the terms *public domain*, *shareware*, and *freeware*. A *public domain* program is a program donated by the author to anyone who wishes to use it. The author gives up part of the copyright of the program and does not charge a fee. The author may retain the part of the copyright that prohibits commercial use or resale of the program. Rarely is any other restriction placed on the distribution or use of the program. A public domain program essentially is free.

Shareware and the inaccurately named *freeware* programs are distributed without a fee. Unlike public domain programs, the authors makes two requests: (1) give the program to as many friends and acquaintances as you wish; (2) make a contribution if you find the program useful or if you use the program in a commercial environment. Resale usually is prohibited. The distribution is free

(hence the misnomer freeware), but the use of the program has a potential charge. You are honor bound, not legally bound, to contribute to a shareware or freeware author for noncommercial use of a product. Commercial users may be legally bound to donate. Either way, those who find the product even marginally useful are asked to send some contribution.

You may be charged a small fee for the disk media when you obtain a public domain, shareware, or freeware product. The fee depends on the source of the program. Appendix B has a further discussion of the public domain, shareware, and freeware stories and lists some sources for these programs.

There is an old saying about these programs: "If you have more time than money, public domain programs are great. If you have more money than time, buy a program that does what you need."

Public domain (often abbreviated as *pd*) and shareware programs can be worth every penny you paid for them (little or nothing); or they can be invaluable tools. These programs carry no warranty, and the documentation is sometimes less than spartan, although most shareware programs have better documentation than public domain programs. The major drawback to public domain, and a potential drawback to shareware, programs is the lack of a publisher—the entity you can call for help or to supply bug corrections. For the most part, you are on your own with these programs.

Don't let the drawbacks dissuade you from using these programs. Many fine "accessories" for DOS are in the public domain or are shareware programs. Many hard disk users rely heavily on some of these tools.

Interacting with QUERY

One interactive program that can be used in batch files is called QUERY.COM, which appeared in the January, 1986, and was corrected in the April, 1986, issues of *PC World*. QUERY is a semi-public domain utility; you can't sell it or buy it. *PC World*, QUERY's publisher, would certainly appreciate your buying the back issues that contain the program, but you may find some other way to acquire QUERY. However you obtain it, QUERY is a tool that can make batch file life easier.

QUERY accepts a message line, which it displays. Immediately after the message you place an "at-sign" (@) followed by the possible one-character responses you want to allow. The program then waits for a keypress from the user. After an allowed keypress,

QUERY exits with an appropriate exit code, which can be tested. If any other key is pressed, QUERY does not respond. A demonstrative batch file is worth a thousand words:

```
.
.
QUERY Do you wish to continue? @YN
IF ERRORLEVEL = 2 GOTO EXIT
IF ERRORLEVEL = 1 GOTO CONTINUE
:CONTINUE
.
.
:EXIT
ECHO BATCH FILE COMPLETE!
```

Note that the ERRORLEVEL numbers are keyed to the kind and order of allowable responses indicated in the line in which the QUERY program is run. In this case, Yes (Y) is 1 and No (N) is 2. Any other keypress will do nothing.

You can set up QUERY with any alphabetic or numeric characters. Although the allowable responses must be indicated by uppercase letters, QUERY will respond to either upper- or lowercase characters when executed. Also, since the IF ERRORLEVEL is actually a greater-than-or-equal comparison, you must compare the higher exit codes first. This little program can add a lot of flexibility to your batch files. Of course, you have to have the program QUERY.COM on the current directory or a directory listed in a PATH statement.

Getting a Number with GETNUM

Another similar public domain program is GETNUM.EXE. Like QUERY, GETNUM accepts input from a user and can be used in a batch file. The difference is that GETNUM only gets a number.

GETNUM's syntax is a simple one:

GETNUM prompt

GETNUM prints the optional prompt and accepts a number. GETNUM then exits using the number it gets as an exit code. A number greater than 255 is accepted as the number modulo 256 (the remainder of number/256). GETNUM asks for a number again if a non-numeric entry or no entry is made (that is, you just press Enter).

You can use GETNUM to modify your numeric-based menu batch file system as follows:

```
ECHO OFF
REM clear the screen, show the choices, and get the choice
:START
CLS
TYPE C:\BIN\MENU.SCR
REM Get the choice and jump accordingly
GETNUM Enter your selection or 0 to quit
IF ERRORLEVEL 6 GOTO TOOHIGH
IF ERRORLEVEL 5 GOTO 5
IF ERRORLEVEL 4 GOTO 4
IF ERRORLEVEL 3 GOTO 3
IF ERRORLEVEL 2 GOTO 2
IF ERRORLEVEL 1 GOTO 1
IF ERRORLEVEL 0 GOTO EXIT
GOTO START
:1
COMMAND /C 1.BAT
GOTO START
:2
COMMAND /C 2.BAT
GOTO START
:3
COMMAND /C 3.BAT
GOTO START
:4
COMMAND /C 4.BAT
GOTO START
:5
COMMAND /C 5.BAT
GOTO START
:TOOHIGH
REM handle too high number
ECHO You selected too high a number
ECHO Please try again
PAUSE
GOTO START
:EXIT
QUERY Do you really wish to exit (Y/N)?  @NY
IF ERRORLEVEL 2 GOTO END
IF ERRORLEVEL 1 GOTO START
:END
```

As before, each program depends on a correspondingly numbered batch file kept in a PATHed directory. You also depend on the MENU.SCR for the text you print. The difference between this and the previous menu batch file is that you never leave this batch file. By using additional copies of COMMAND.COM, you invoke the batch files that run the individual programs. When the programs and the subsequent batch files quit, the additional copy of COMMAND.COM quits. You return to the menu batch file, which jumps back to the start and does the work over again.

This batch file is more robust than the previous files and serves as an excellent model. There is one gap, however, that neither QUERY or GETNUM bridges: the batch file cannot accept interactive words such as a disk drive designation or a file name. The next example bridges this gap.

Using the Environment in Batch File Technique

A third public domain program is ANSWER.COM. ANSWER works with PC DOS V3.1 and V3.2, not V2 or V3.0. ANSWER works with some versions of MS-DOS V3. The reason ANSWER works on some versions of DOS is that ANSWER relies on an undocumented feature of DOS V3.

ANSWER accepts characters from the keyboard and places the answer in something called the *environment*. The environment, briefly mentioned in the SHELL command section in Chapter 4, is a concept DOS has inherited from the UNIX world. In simple terms, DOS reserves an area in RAM for strings, a "safe haven" for some text. The text (strings) may be recalled by your programs and by DOS.

Like PROMPT's prompt-string and IF's *string==string*, environmental strings hold simple text. To see an example of environmental strings, at the DOS system prompt type

SET

You'll see a list of lines something like this:

```
COMSPEC=C:\COMMAND.COM
PATH=C:\BIN;C:\
PROMPT=$p$_yes, Don? $
```

The lines you see probably will be different. Notice a similarity in the lines. Each line starts with an uppercase word, followed by an equal sign (=), followed by some text.

The uppercase word beginning the line is the *environmental variable*. The text following the equal sign is the *environmental string*. In the preceding example, COMSPEC is the environmental variable for the full file specification of the command processor. This variable is established by DOS when you start your system. COMSPEC is used by certain programs and DOS to locate the command processor.

DOS automatically sets COMSPEC. You set the PATH and PROMPT variables using the PATH and PROMPT commands. You also can set these and other variables by using the SET command. The syntax of SET is

SET *variable_name=string*

Typing SET by itself shows the current environmental variables. Using **SET** *variable_name=string* sets *variable_name* to whatever is contained in *string*. Using **SET** *variable_name=* (not giving a string) clears the variable from the environment.

Many programs use the safe haven of the environment to hold start-up or operational information. An example of environment searching programs is Brief™, a popular programmer's text editor. Brief uses two environmental variables for the location of its help files (BHELP) and auxiliary files (BPATH). To set the variables, use the following SET commands:

SET BHELP=c:\brief\help
SET BPATH=c:\brief\macros

When you start Brief, the program looks for the variables, examines the associated strings, and automatically looks in C:\BRIEF\HELP for the help files and C:\BRIEF\MACROS for the auxiliary files. When you include these SET statements in your AUTOEXEC.BAT file, DOS automatically includes these variables in the environment. Then, when you use Brief, you don't need to give the SET commands each time. The information is safely tucked in the environment, ready for use at any time. Brief simply examines the environment and adapts itself to the information.

ANSWER is another program that uses the environment. Unlike Brief, which only reads the environment, ANSWER places items in the environment. The syntax of ANSWER is

ANSWER *optional_prompt*

ANSWER waits for a line to be entered from the keyboard. If the optional prompt is given, ANSWER displays the prompt first. The line entered by the user is placed in an environmental variable called ANSWER. For example, the following batch file line invokes ANSWER:

ANSWER Enter the name of the source file:

ANSWER displays the prompt and waits for the user to enter some information. When the user presses Enter, ANSWER places the entered information in the environment.

The following sample dialogue uses the preceding line in a batch file named TEST. First, run the batch file by entering its name at the system prompt.

```
TEST
ANSWER Enter the name of the source file:
Enter the name of the source file:TEST.BAT
```

By entering *TEST.BAT*, you instruct ANSWER to place that string in the environmental variable called ANSWER. To check to see if that happened, issue the SET command.

```
SET
COMSPEC=C:\COMMAND.COM
PATH=C:\BIN;C:\
PROMPT=$p$_yes, Don? $
ANSWER=TEST.BAT
```

As you see from the test, ANSWER places the response (TEST.BAT) in the environmental variable ANSWER. ANSWER leaves an exit code—0 if successful and 1 if ANSWER cannot place the string in the environment. You test for a failure by using an IF ERRORLEVEL 1 statement.

Here is the DOS trick: In PC DOS V3.1 and V3.2, and MS-DOS V3.1, you can use an environmental variable in a batch file by placing the percent sign (%) before and after the variable's name. This quirk was documented in DOS V2.1, but works in the mentioned versions of DOS. By using the %name% technique, you can improve DOS's interaction with users.

The following modified version of the menu system also includes a help function:

```
ECHO OFF
REM clear the screen, show the choices, and get the choice
:START
CLS
TYPE C:\BIN\MENU.SCR
REM Get the choice and jump accordingly
ANSWER Enter your selection or 0 for help
IF ERRORLEVEL 1 GOTO OOPS
IF %ANSWER%. == . GOTO START
IF %ANSWER%. == 0. GOTO HELP
IF EXISTS C:\BIN\%ANSWER%.BAT GOTO OKAY
ECHO Sorry, your choice does not exist.
PAUSE
GOTO START
:OKAY
COMMAND /C %ANSWER%
GOTO START
:HELP
ECHO Enter the number of the program you want help on,
ANSWER or just hit Enter to see a list of files
IF ERRORLEVEL 1 GOTO OOPS
IF %ANSWER%. == . GOTO LIST
IF %ANSWER%. == 0. GOTO LIST
IF EXISTS C:\HELP\%ANSWER%.HLP GOTO HOKAY
:SORRY
ECHO SORRY, HELP DOES NOT EXIST FOR THIS NUMBER
:LIST
ECHO HERE ARE THE AVAILABLE HELP FILES --
DIR C:\HELP\*.HLP /W
GOTO HELP
:HOKAY
TYPE C:\HELP\%ANSWER%.HLP | MORE
PAUSE
GOTO START
:OOPS
ECHO I have an internal problem. Contact your system
ECHO supervisor.
```

This batch file is similar to the earlier versions. Notice the lines

```
TYPE C:\HELP\%ANSWER%.HLP | MORE
COMMAND /C %ANSWER%
```

These lines type any help file or run any batch file. More than 20 lines have been eliminated from the previous menu batch file by using ANSWER. Notice that ANSWER allows a blank entry. You can use . == . to catch blank entries before DOS issues its dreaded `Syntax error` for comparing a null string.

If ANSWER sets an error level of 1, the system is out of space for the environment or the strings in the environment have been corrupted. If this happens, an error message is displayed and you fly out of the batch file, because nothing can be done within the batch file. The message asks you to contact a knowledgeable person to correct the environment problem. The steps to prevent this problem are explained in a later section of this chapter.

The following batch file does an interactive COPY using ANSWER and the DOS SET command to store more than one ANSWER in memory:

```
ECHO OFF
CLS
:START
ANSWER Enter the full name of the source file -
IF ERRORLEVEL 1 GOTO OOPS1
IF %ANSWER%. == . GOTO OOPS2
SET SFILE=%ANSWER%
ANSWER Enter the destination -
IF ERRORLEVEL 1 GOTO OOPS1
SET DFILE=%ANSWER
COPY %SFILE% %DFILE% /V >NUL
GOTO EXIT
:OOPS1
ECHO I have an internal problem. Contact your system
ECHO supervisor.
GOTO EXIT
:OOPS2
ECHO I need some source file name!
ECHO Please try the command again.
GOTO START
:EXIT
SET ANSWER=
SET DFILE=
SET SFILE=
```

The batch file is fairly straightforward. One highlight is the use of SET to place the result of ANSWER (**%ANSWER%**) into another environmental variable. The lines

```
SET SFILE=%ANSWER%
SET DFILE=%ANSWER%
```

copy the contents of the variable ANSWER into the SFILE (source file) and DFILE (destination file) variables.

Another highlight is using I/O redirection to pipe the output of the COPY command into the null device (>NUL). This redirection sends the output of COPY into the bit-bucket, so the `file(s) copied` message will not be shown. This technique is useful to obtain a "quiet" (uncluttered) screen when using some DOS commands.

The final three lines free the environmental variables ANSWER, DFILE, and SFILE. Other environmental variables can reuse this space. It is good practice to include this step because too many "lingering" environmental variables can exhaust string space and cause batch files to "bomb."

If ANSWER returns an exit code of 1, environment space is exhausted or the environment space is corrupt. The environment is seldom corrupted; only flawed programs corrupt the environment. More frequently, you simply run out of environment space.

A two-fold process prevents space exhaustion. First, always free the temporary environmental variables used in batch files. If you do not reuse these names between the running of batch files, the variables stay in memory. DOS is not a good string housekeeper; unwanted strings are not automatically discarded. Therefore, failing to free temporary variables quickly uses up the string space.

The second step is to increase the size of the environment. The SHELL command of CONFIG.SYS should be used. DOS V2 through V3.1 default to 128 bytes of environment space; DOS V3.2 defaults to 160 bytes. A safer environment size is between 320 and 480 bytes, more than sufficient for most systems. If you have not given the SHELL command, look in Chapter 4 for the directions on adding this command to your CONFIG.SYS file.

Some circulating public domain programs also increase the size of the environment. You might use one of these programs as the alternative to using the SHELL command. Check the sources in Appendix B for one of these programs.

A Summary of Interactive Batch File Techniques

DOS does not provide a method for the user to interact with a running batch file. You must use outside programs to get this feature. These outside programs make batch files more versatile and powerful.

Interactive batch files can be fun to use. Batch files seldom execute as quickly as programs, however. Menu and windowing programs, which are discussed in Chapter 14, run faster and have more features. Menu and windowing programs also have faster, more comprehensive error handling.

Only your imagination limits what batch files can do. Read Appendix B on public domain and shareware programs; get copies of the programs; then experiment. If in doubt, build test batch files using the DIR command, which cannot "hurt" files. Once you master the auxiliary programs, your batch files will be more powerful and useful.

Remember that the *%name%* technique is undocumented and does not work with all versions of DOS. The facility is here today (if you have the right version of DOS), but may be gone tomorrow. If this facility is not in later versions of DOS, *do not blame* Microsoft or anyone else. That's the risk in using undocumented features. The onus of support is on you, not the publisher. But the rewards (using interactive batch files with ANSWER) can be well worth the risk (reworking the batch files later).

Summary

In this chapter you explored advanced batch file technique. You also learned how to use additional DOS commands and some outside programs. The important points of this chapter for you to remember are

- Use absolute drive and path names within batch files if the batch file can be run from any directory.
- Batch files can run other batch files. The normal direction of control is from the first batch file to the second, meaning that only the last line of a batch file should invoke the second batch file.

- You can approximate a GOSUB batch file command by invoking additional copies of the command processor in the form **COMMAND** */C command.*
- The PROMPT command alters the information presented by the DOS system prompt.
- The AUTOEXEC.BAT file can be used to invoke commonly used background programs or utilities such as RAM disk and print-spooler software. However, do not invoke these programs if your applications programs routinely need the RAM space used by the auxiliary programs.
- Interactive batch files can be constructed by using public domain or shareware programs.
- The SET command can place strings into the environment. In later versions of DOS, you can use the contents of the environmental variables in your batch files through the *%name%* technique (placing a % before and after the environmental variable name).

The following chapter summarizes the commands and techniques presented and gives more definite guidelines for organizing your hard disk.

Chapter 9

Guidelines for Organizing Directories

This chapter contains some general techniques and approaches for organizing your hard disk and begins to bring together what you have already learned. You might say that this chapter ties the first knot around what has been presented.

You have learned some common and uncommon techniques with batch files and other DOS subjects. The goal for the remainder of this book is to make computer operations easier and faster—a goal agreeable to all. To achieve this purpose, many specifics (and some generalities) can be given. To reach this optimal state, you must do some necessary work, but that work will be amply rewarded.

This book cannot provide exact commands or layouts for every person. No setup is universal or even good for everyone. Each person must use the hybrid setup that best applies to the programs used, the hard disk management tools available, and the temperament and expertise of those using the computer.

This chapter gives guidelines—none truly absolute. Feel free to ignore the guidelines, but remember one simple admonishment: Have a reason to ignore what is said. I have always looked for the easiest way, the fastest way. The information presented is based on my own experience. You may find different ways to organize. Simply prove to yourself that your way works better.

Initial Directory Organization: The Root and DOS

Given the powerful nature of a hierarchical directory system, it is an operational sin not to use the directory structure. For that matter, use of a hard disk without a logical subdirectory structure is a crime punishable by delays and befuddlement.

One underlying guideline is that subdirectories should hold as few files as possible. If you have too many files, create another subdirectory and divide the files between the new and old directories. This philosophy can easily be overused, but I follow that guideline more often than I violate it.

The second underlying philosophy is to divide and conquer. Place major software systems in their own directories. The division is either by purpose (word processing, spreadsheet, graphics) or by applications program (WordStar, 1-2-3, PC Storyboard). As discussed in Chapter 5, hybrids of the two ways of organization often work better. You may organize by purpose and then by program or by program and then by purpose.

Starting at the Top

Start with the root directory. DOS and some programs enforce certain limitations; therefore, some files must be located in the root. Because all roads lead to the root, the goal is to keep the absolute minimum number of files in the root.

CONFIG.SYS and AUTOEXEC.BAT must be in the root directory. The MIO.SYS and MSDOS.SYS files (or IBMBIO.COM and IBMDOS.COM) also are located in the root. These files, however, are hidden and do not "appear" in the directory. Although COMMAND.COM can be moved to another location (with the CONFIG.SYS SHELL command), leave this file in the root directory. Later, you will learn two defensive steps to ensure that COMMAND.COM is not inadvertently changed or erased.

An absolute annoyance is placing all DOS files in the root directory. The root is your home directory. The remaining directory structure starts in the root. Wading through fifty-some files just to find a couple of subdirectory entries is a waste. To ease the burden on your memory and on DOS, place DOS files in their own directory.

Creating a Level-One Subdirectory

Chapter 5 explains how to establish a level-one subdirectory—a child subdirectory of the root—to hold DOS files. The suggested names for this subdirectory are BIN or DOS. The name BIN is inherited from the UNIX world where BIN is the subdirectory that holds most of the commonly used UNIX software. Because many essential auxiliary programs not provided with DOS go into this directory, the word BIN has significance. The origin of the name DOS is self-evident. If you choose the name DOS, you should put your non-DOS auxiliary programs into their own directory called UTILS.

My personal preference is DOS, but some associates insist on BIN. For convenience, I use BIN in this book. You can use either name; just make the choice and stick with it. Because you will later create many batch files, the path name must be fixed. If you choose DOS, remember to substitute the name DOS or UTILS where I say BIN.

In Chapter 4, you are instructed to copy both DOS diskettes to BIN. The files holding the BASIC language and the sample BASIC programs are also copied to BIN. The sample BASIC programs are not essential and so can be moved from BIN (or DOS) to another directory. But before moving these files, ask yourself one question: Will I develop many of my own BASIC programs? If the answer is yes, place the sample BASIC programs in their own subdirectory. If the answer is no, you will still need to place the program files in their own subdirectory. The only difference is which subdirectory.

In either case, create a level-one subdirectory called BASIC. The command is

```
MKDIR \BASIC
```

If you don't plan on developing many BASIC programs, move the sample BASIC files to this directory. The commands are

```
COPY \BIN\*.BAS \BASIC /V
ERASE \BIN\*.BAS
```

If you plan to develop BASIC programs, you need to segregate the sample programs from the programs you develop. Create another level-two directory called SAMPLES with this command:

```
MKDIR \BASIC\SAMPLES
```

Move the BASIC programs using these commands:

```
COPY \BIN\*.BAS \BASIC\SAMPLES /V
ERASE \BIN\*.BAS
```

The BASIC files are out of the way.

If you rarely use BASIC, you might copy the BASIC language files themselves into \BASIC. The exact names of the files holding BASIC vary. IBM uses BASIC.COM and BASICA.COM. COMPAQ BASIC has three files: BASIC.COM, BASICA.COM, and BASIC.EXE. Some MS-DOS systems use GWBASIC.EXE. However, I suggest taking this step later, after you have developed a feel for the kinds of things you will do with the computer.

You'll soon begin accumulating utilities programs and other DOS tools. These programs will come from the public domain, computer magazines, purchases, "roll-your-own" (self-written), and other sources. You can place utilities programs in the \UTILS or the \BIN subdirectory. If you like the philosophy of the DOS subdirectory, create a level-one subdirectory called UTILS. Place these programs here. If you're a \BIN fan, just copy these files to \BIN.

Programs That Use Subdirectories

The next major step is to examine the applications programs you use: spreadsheets, word processors, database managers, graphics programs, communications, accounting, and any of the myriad other kinds of programs that process your data. Your first impulse may be to copy all the programs into one directory and put all your data files into the same directory. I've already explained why that approach is less than optimal.

Before you can part the electronic waters of the program and data files, you need to perform some detective work. First, you examine the program and its own files. Then look at the data files you create and use. Ask yourself the following questions about each program:

1. Does the program support hierarchical directories?
2. How extensive and congenial is that support?
3. If the program consists of several files, does the program know where to locate its own files?
4. For data files, can data files be located anywhere?
5. Can you conveniently use data files located in other directories?

Program Files

Your investigation starts with the program itself. When you buy a program, you may find more than one file on the diskette. Many programs consist of several files. Disregarding demonstration files, the remaining files fall into four categories: the main or master program, program overlays, auxiliary programs, and auxiliary data files.

If the program is contained in a single file, that file is the program. In larger applications programs, the program you invoke is considered the *main* or *master program*. The main program may be a simple menu or may contain many of the program's functions.

The multiple-piece program brings additional program parts one at a time from a file into memory. The program uses a part and then discards that part. The file or files containing the additional parts are *program overlays*. These files overlay portions of the resident program in memory. For example, not all the WordStar functions are in WS.EXE, the main WordStar file. The overlays are in a file named WSOVLY1.OVR. When WordStar needs some function, such as the part that prints documents, the program loads the overlay into memory.

Another technique used by programs is to segregate their functions in modules, each of which is a separate program. For example, 1-2-3 has a menu program, LOTUS.COM; a graph-printing program, PGRAPH.EXE; the 1-2-3 program itself, 123.EXE; the installation program, INSTALL.EXE; and other programs that convert text or data files to worksheet files and back. These additional programs are considered *auxiliary programs*. Auxiliary programs are also provided by programs like spelling checkers and correctors, which use separate programs to count words or show a frequency distribution for the words in a document.

The difference between the two types of auxiliary programs—those that come with 1-2-3 compared to a spelling checker, for example—is that one set can be invoked from a menu. 1-2-3 is this type. The other type—most spelling checkers, for instance—instructs you to run the auxiliary programs separately. Functionally, the two are no different. Because the auxiliary programs are to be used with files created by the main program, I generally keep the auxiliary programs with the main programs.

Auxiliary data files are data files used exclusively by the program, usually for configuration information or to house the text for help messages. The texts of 1-2-3 and WordStar help messages are located in data files. 1-2-3 driver files for your keyboard, display, and printer

are in another data file. A program data file is provided by the manufacturer or created by an installation program for exclusive use by that applications program.

For this discussion, I classify any file that is not the main program as an *auxiliary file*. This term applies to overlays, auxiliary programs, and auxiliary data files. When discussing a program's data files, I use the term *auxiliary data files*. When discussing the files you create, edit, use, or destroy, I use the term *data file*.

With the definitions established, you are ready to examine the program. Ask the following questions:

1. Does the main program know where its auxiliary files are located?
2. Can you place these files anywhere, or must the files be in the current directory?

Programs written for DOS V1 are completely ignorant about subdirectories. These programs may be so obsolete that the programs handle only two disk drives: A and B. For these antiques, your best bet is to place all the files in one subdirectory. You may need to resort to one of the DOS trickery commands, such as ASSIGN, SUBST, or JOIN. The ASSIGN and SUBST commands are discussed in this chapter.

Many programs written before DOS V3 must have their auxiliary files in the current directory. This fact means that you are almost forced to copy all files for the program into one directory and run these programs from that directory. However, the SUBST command of DOS V3.1 and later may be useful in overcoming the "one-directory-fits-all" limitation.

Many programs published for DOS V3 and later provide a special installation option through which you can inform the program of the subdirectory holding the program's auxiliary files. These programs are amenable to being installed in any directory (or more than one directory) and to being used from any directory.

Philosophically and operationally, the sensible way is to place most or all program files for an application (like 1-2-3) in that program's own subdirectory. Most publishers include a convenient installation batch file or program that creates the separate subdirectory (or subdirectories) and copies the relevant files.

Sometimes no installation program or batch file is included; the manual just tells you to copy all the files to your hard disk. You can create the subdirectory and copy the files yourself. To do so, create a subdirectory, usually a subdirectory of the root; make the new

subdirectory the current directory (CD to the directory); and copy the files from the floppy diskette to the hard disk.

Generally, all the program files are copied to one directory. But in some cases, creating additional directories for second and third groups of programs is helpful. For example, the graphics-character fonts for 1-2-3 can be placed in their own directory. This technique is demonstrated in the model hierarchical directory in Appendix A. The font files are held in the directory \123\FONTS. Many users follow the "out-of-sight, out-of-mind" philosophy. 1-2-3 uses the fonts; you do not, directly. Moving the fonts helps decrease clutter in your 123 directory.

Remember that, based on the way programs handle their own files, they can be classified into three categories:

1. Programs that work within one and only one directory
2. Programs that can be tricked into working with more than one directory
3. Programs intelligent enough to work with any directory

You'll return to setting up a program in your directories after examining how well the program works with different directories for user-created data files.

Data Files

Based on how the program operates with subdirectories and data files, programs fall into four categories:

1. Programs that know nothing about subdirectories
2. Programs that can be tricked into using separate subdirectories
3. Programs that can use separate subdirectories for data files
4. Programs that conveniently use separate data file subdirectories.

These differences are important.

You may assume that the easiest route for handling data files (not auxiliary data files) is to create or use these files in the same directory with the program. This assumption has merits and drawbacks. The merit of using one directory for both program and data files is that you can go into the directory, start the program, and know that all data files for use with the program are in this single

directory. You change directories only once. Your programs need not support hierarchical directories. You need not remember any additional directories that hold your data files.

The drawbacks of having "all your eggs in one basket" are the increased chance of destroying a program file; increased housekeeping burden (copying, deleting, backing up, and other activities); and the inability to group data files logically.

Routinely creating, editing, and deleting data files in the same directory with the program files increases the chance of erasing or overwriting a valuable program file. Yes, data files themselves are valuable, but many data files can be "dead storage" without the appropriate program files to use the data. If your program looks at (uses) a different directory for data files, you reduce the possibility of the devastation of an inadvertent ERASE *.*.

Housekeeping tasks—such as getting a directory and copying, erasing, or backing up files—are less convenient when all your files are in the same directory. When many programs display a list of files in the current directory, you must wade through the list and separate your program files from your data files. Selective copying, erasing, or backup tasks require more time; either more files are copied or several commands using different file names must be issued.

Finally, without subdirectories you lose the important capability to organize files by logical working groups. Grouping files in separate subdirectories has great merit. For example, word-processing documents can be grouped by working types (LETTERS, PROPOSAL, and CONTRACT); receivers of the document (CONTACTS, SUPPLIER, and FRIENDS); or author (MARGARET, DOTTIE, KATHY). Each group can be placed in a separate subdirectory. The entire set of subdirectories can be children (subdirectories) of \WORDS or \DOCS.

Given absolute freedom, the preferred choice is to establish all data files in separate directories. In real life, however, that absolute freedom does not exist, and this approach can be difficult or impractical to implement. Some programs, for example, do not allow path names. Again, WordStar V3 comes to mind. Written before hard disks were popular, WordStar allows disk drive and file names but not path names. You must work outside of WordStar to edit files in different directories.

On the other hand, 1-2-3 allows you to specify a default subdirectory name for worksheets. Placing worksheets in a subdirectory separate from the program is easy. Changing subdirectories is less than convenient with 1-2-3 Release 1A,

however. You must maneuver through the menu system to the Change Default Directory option and then set a new default directory for worksheets. 1-2-3 Release 2 is more gracious. You may use a subdirectory name at any place 1-2-3 accepts a file name. Therefore, using a set of directories to hold worksheets is more convenient with 1-2-3 Release 2.

When you have only a few data files, the hassle factor of creating and maintaining separate subdirectories is large. If fewer than a dozen or so data files are involved, little convenience is gained by using a separate subdirectory for the data files. However, the safety of segregating the data from program files may outweigh the inconvenience.

Subdirectory Decisions

A program may act graciously toward placing various files in different directories or may make putting files into subdirectories difficult to downright impossible without some DOS trickery. The degree of difficulty can be the same or different for the program's auxiliary files and the program's data files. The degree of difficulty and the number of data files that will be used should determine your hierarchical directory setup.

Regardless of the number of data and program files, setting up any subdirectory system follows a definite order. To make the procedure clear, I take you through the steps I followed when I created the model disk directory shown in Appendix A. These steps are summarized as follows:

1. Make a subdirectory for the program itself.
2. Move unneeded files out of this directory to another directory.
3. Determine which sets of auxiliary files can be moved into separate subdirectories and place the files in the appropriate directories.

Step 1. Make a subdirectory for the program itself.

The first step in the process is universal. Usually, the directory for the program itself is a level-one subdirectory, a child of the root. Copy the program and auxiliary files to this directory. In the model disk directory in Appendix A, 1-2-3 has a separate directory. I ran the installation program and specified *\123* as the target directory.

Step 2. Move unneeded files out of this directory to another directory.

The second step is also universal. You need to decide which files are necessary in the program directory and which are not. When creating the model, I decided to keep a copy of the 1-2-3 installation files on the disk. Because these files are used very infrequently, I copied them to a separate subdirectory. However, 1-2-3 auxiliary files must be in the current directory when 1-2-3 runs. If I change the configuration by rerunning the installation program, the resulting configuration file (123.CNF) must be copied back to the 123 directory.

Step 3. Determine which sets of auxiliary files can be moved into separate subdirectories and place the files in the appropriate directories.

After reading the 1-2-3 documentation, I discovered that I could move the graphics fonts to a separate directory. I created the directory FONTS and moved the FNT (font) files to this directory.

I also found that my graph files could be placed in a separate directory. I created the directory, ran PrintGraph program, and installed the name C:\123\GRAPHS as the subdirectory for my graphs.

1-2-3 also allows worksheets in separate directories, so I created subdirectories for major groups of worksheets. Although Release 1A is more clumsy about changing directories, this approach is appropriate regardless of the 1-2-3 release. I'd use a single directory called SHEETS, however, if I anticipated creating fewer than 10 to 15 worksheets. Using additional directories for so few files causes more work than benefits.

Batch Files To Start Programs

Because 1-2-3 requires certain files in the current directory, the best operational strategy is to pivot off the 123 directory. I write a batch file, which is placed in a directory specified by the PATH command (a PATH directory). This batch file moves me to the 123 directory and invokes the program. The batch file, appropriately entitled 123.BAT, is

```
C:
CD \123
LOTUS
```

The batch file ensures that C: is the current disk drive, moves to the 123 subdirectory, and then invokes the LOTUS menu. If you're proficient with 1-2-3, you can change the last line to

123

This statement directly invokes the spreadsheet program of 1-2-3, bypassing the Lotus menu.

WordStar 2000 is similarly gracious about using document files in any directory but is ingracious about running the program from any directory. Therefore, I use the same pivotal strategy to move into the directory holding WordStar 2000 and then invoke the program. I change paths while in WordStar 2000 or specify the path name with the file name when I am editing individual documents.

Programs That Use Subdirectories Unwillingly

The preceding example for 1-2-3 works well for programs that recognize and conveniently use subdirectories. What about programs that almost force all files into the current directory? For example, WordStar V3 is ingracious about using any subdirectory. The solution involves some DOS trickery, the SUBST command.

Tricking Programs by Using SUBST

SUBST is an external (disk-based) command in DOS V3.1 and later. The command "tricks" your programs (and DOS) into treating a subdirectory as if it were a disk drive. You create an alias for the subdirectory by using the SUBST command. The general form for creating an alias is

*dc:pathc***SUBST dp: dr:pathr**

The *dc:pathc*\ represents the disk drive and path holding the SUBST command. If SUBST is on the path set by the PATH command, you can omit this portion.

SUBST accepts two parameters when you are creating the alias. The first parameter (*dp:*) is the name of the alias disk drive. The second parameter (*dr:pathr*) is the actual drive and path name that *dp:* will represent.

For example, to disguise the subdirectory C:\LETTERS\BUSINESS as drive E, you type

SUBST E: C:\LETTERS\BUSINESS

Then, whenever you specify drive E, your programs and DOS actually use the subdirectory C:\LETTERS\BUSINESS.

You can use any drive name for the alias, but with four caveats:

1. Don't give a real disk drive name for the alias. If you do, you cannot use that disk drive until the substitution is broken.
2. Although using a "high" disk drive letter has merit, a drive name higher than E requires that you use the LASTDRIVE command in the CONFIG.SYS file. This command was covered in Chapter 4.
3. Use absolute path names. Relative path names cause problems.
4. Some DOS commands do not like being tricked. For example, CHKDSK will not analyze the substituted disk drive (E:). CHKDSK will display an error message and quit. Other DOS commands will not carry your command to that letter. Writing a volume label with the LABEL command to drive E will label the underlying physical drive instead, drive C. When tricking DOS, be careful of the commands you use.

To see a list of the substitutions you have made, type just the command, **SUBST**. A list of disk drive names and their corresponding real disk drives and paths is displayed. If no substitutions are active, SUBST displays nothing.

To break the substitution, type

dc:pathc\SUBST dp: /D

Give the alias disk drive name (*dp:*) with the */D* (disconnect) switch. As with some other DOS commands, SUBST does not display any message when DOS has successfully disconnected the alias drive name from the real name.

Choosing SUBST for the Data Files or the Program

You have two options for SUBST with WordStar. You can substitute for a data file subdirectory or for WordStar's auxiliary files. In the WordStar installation program, you can specify which disk drive holds the WordStar overlay (auxiliary) files. Therefore, you can install an alias disk drive name in WordStar and use SUBST to trick the program into finding the WordStar files.

For example, having used the LASTDRIVE=G command in CONFIG.SYS, you can tell WordStar that its files are located on drive G. Copy the WordStar files to a subdirectory called \WORDSTAR. Next, you move to the subdirectory holding the files to edit and use the following batch file (called WS.BAT and placed in a PATH directory) to invoke the program:

```
SUBST G: C:\WORDSTAR
C:\WORDSTAR\WS %1
SUBST G: /D
```

The batch file first establishes the directory C:\WORDSTAR as the substituted disk drive G. Next, the file invokes the word-processing program WS.EXE. After you exit from WordStar, the batch file disconnects the substitution. Disconnecting is good practice. You may want to use this technique with other programs using the same disk drive name.

Notice the **%1** parameter in the batch file. WordStar automatically loads and moves to editing in the document mode if you supply a file name on the command line. Suppose that you type

```
WS MANAGE.TXT
```

WordStar comes up in the document editing mode in the file MANAGE.TXT. If you omit the file name, the *%1* is null. WordStar does not get any parameter and displays the menu.

An alternative is using SUBST to trick WordStar into seeing your files on a separate disk drive. You need not "install" an alias disk drive name in WordStar. This batch file accepts a parameter also. The parameter is the name of the subdirectory that holds your document files.

```
IF %1. == . GOTO JUMP
SUBST G: C:%1
G:
:JUMP
C: WS
IF %1. == . GOTO END
C:
SUBST G: /D
:EXIT
```

When invoking the batch file with a path name, a SUBST command establishes the named subdirectory (which holds your data files) as drive G. G: becomes the current disk drive. WordStar is then invoked and happily edits and prints your documents on the phony disk drive.

disk drive. When WordStar exits, you change back to drive C and the substitution is broken. Notice the order of the last two steps. This order is important. DOS can act ingraciously to having a nonexistent disk drive as the current disk drive.

The batch file also uses the check for a null parameter. If no parameters are given, no substitution is made and the current disk drive is not changed. This allows you to edit files on a floppy diskette. You simply invoke the batch file, then specify within WordStar the name of the disk drive containing your document files or change the current disk drive to the floppy disk drive.

This technique can be used for any program that accepts disk drive names. You substitute the appropriate directory for an alias disk drive name. If the program allows you to "install" its home disk drive location, you can give an alias drive name and use SUBST to trick the program into finding its auxiliary files. In this case, you move to the directory holding your data files and invoke the batch file. The first approach pivots off the program's home directory and substitutes an alias disk drive name for the data file's directory.

One more technique is possible in coping with programs that cannot locate their own auxiliary files. The solution, a data file PATH command, is covered in Chapter 6.

Programs That Know Their Own

The final category is programs that know where their auxiliary files are and allow data files to be located anywhere. For example, the path name for the directory holding the auxiliary files can be installed in NewWord, a WordStar clone. NewWord also allows you to change the logged (current) disk drive and directory and to give path names with file names. Your options are wide open with NewWord.

After you install NewWord into its own directory (\NW), the batch file to invoke the program (NW.EXE) is simply

```
C:\NW\NW %1
```

Because NewWord knows where its files are located, you can invoke NewWord from any directory. The batch file invokes NewWord. As in the WordStar example, you can give an optional file name. NewWord automatically brings up that file in the document editing mode.

This technique is useful with any program that knows where its own files are located and can conveniently change data file

subdirectories within the program. You simply change to whatever directory you wish and invoke the program.

If you don't mind the additional step, you can start NewWord from any directory and then specify path names with your file names or change the default directory to the proper directory holding the documents. If you have a quick directory changer, such as GO, you may prefer changing directories outside of the program.

Unyielding Programs

Some applications programs are unyielding in their disk drive assignments. These programs *require* that the program be run from one disk drive, usually drive A; and the data files must reside on another specific drive, usually drive B. These are simply antique programs that assume that the only disk drives you have are floppy disk drives.

Tricking Programs with ASSIGN

For these antiques, you can use another DOS trickery command: ASSIGN, an external command that makes one disk drive "look like" another. Just as SUBST allows you to reroute to a specific subdirectory any activity intended for a disk drive, ASSIGN allows you to reroute activity from one disk drive to another drive. The syntax is

*dc:pathc***ASSIGN** *ds1=dc1 ds2=dc2* . . .

The *ds1* and *ds2* are the disk drives to be rerouted. The *dc1* and *dc2* are the disk drives that DOS will use instead of *ds1* and *ds2*, respectively.

ASSIGN does not allow path names, but you can make several assignments at once. Just be sure to separate the assignments with spaces (or commas for DOS V2). For example, to assign both drives A and B to drive C, you use

ASSIGN A=C B=C

Whenever DOS or your programs attempt to use drive A or B, the activity actually takes place on drive C. After copying the needed program and data files to a subdirectory, change into that subdirectory and use the following type of batch file to trick the program:

```
C:
CD \PROGRAM
ASSIGN A=C B=C
PROGRAM
ASSIGN
```

This batch file changes to the correct directory, assigns the disk drives, invokes the program, then cancels the assignments. When you issue the ASSIGN command without parameters, all effects of ASSIGN are canceled.

Another use for ASSIGN is the wholesale rerouting of a disk's activity to a RAM disk. That technique is discussed in Chapter 12.

ASSIGN's effects are similar to those of SUBST. Although DOS is fooled by an assignment, some commands do not go along with the rerouting willingly, and the results with these commands can be unexpected or messy. ASSIGN and SUBST cannot fool all applications programs. Copy-protection schemes or other nonstandard quirks may force you to work from the floppy diskette.

Remember that ASSIGN can be used for wholesale rerouting of activity from one disk drive to another. Given a choice for DOS V3.1 and later, use SUBST rather than ASSIGN. Always break the assignment or substitution after you are finished. Also consider either command an over-the-counter computer drug: use only when needed.

Tricking DOS V2

SUBST is provided with DOS V3.1 and later. If you are using DOS V2, you cannot use batch files to run programs from anywhere. DOS V2 users also cannot use a path name when running a program or a batch file. But you can use the following trick to get some of the efficiency of these tips.

Copy the start-up program to a directory specified by PATH. For instance, copy WS.EXE to UTILS or BIN. Now, the major start-up file can be invoked from anywhere. Leave the program's auxiliary files in their own directory. You then must use one of the data path programs given in Chapter 13 to trick the programs into finding their auxiliary files. When you have a data path program in place, move to the directory holding your data files and invoke the program. PATH will locate the main program, and the data path program will help the main program find its auxiliary files. You get the same advantages.

Auxiliary Programs

Many programs are available to supplement other programs. Word-processing programs have spelling checkers and correctors; spreadsheets have auditors and sideways-printing programs; many companies publish programs that make DOS easier to use or extend its features. I call these types of programs *auxiliary programs* or *tools*. Like the hammer or screwdriver, these additional programs make other programs, including DOS, easier or more functional to use.

Where should you place these programs?

My answer follows almost the same guidelines as for program and data files:

1. Does the program know where to get its own auxiliary files?
2. Can you use data files in different subdirectories?
3. How many files make up the auxiliary tool?

The question of location has no right or wrong answer. For instance, one package of auxiliary tools for DOS is the Norton Utilities. Version 3 has 25 files, 20 of which are programs. Each program is independent and does not need an auxiliary file. Because these programs are frequently used, my inclination is to place the Norton program files in my \UTILS or \BIN directory. However, I seldom use several of these programs, so \UTILS or \BIN is burdened with some files that are mostly dead weight. One alternative is to place the entire set of Norton files in their own directory, \UTILS\NORTON or \BIN\NORTON, and add the directory name to the PATH command. Using the latter scheme, all the programs are available for use without requiring a path name.

MicroSpell, a spelling checker and corrector, has about a dozen files. My inclination is to place MicroSpell and its files in a separate subdirectory. Because the location of MicroSpell dictionaries can be placed in an environmental variable (appropriately named LEX, the name of its dictionary files), the choice of a separate directory is beneficial.

Some programs are self-contained. Utilities like GREP, a more comprehensive FIND command, consist of a single program. These types of files are best relegated to the \UTILS or \BIN directory.

Having enough complete sets of auxiliary tools is reason to realign directories. For instance, much of my time is spent writing. I have several sets of auxiliary word-processing tools: spelling checkers, grammar and style checkers, word frequency programs, and others. Because these programs know where their files are located, I have created a level-one subdirectory called WORDS. My word processor is in its own level-two subdirectory. My spelling checker also has its own subdirectory, \WORDS\SPELL. My style checker has another subdirectory, \WORDS\RIGHT. I create batch files that are placed in a PATH directory—\UTILS or \BIN—and I can invoke these programs from any subdirectory.

I also split my documents among several directories within WORDS, such as LETTERS, MEMOS, and MHD (the subdirectory holding this book). Because my programs can be invoked from any directory, I simply move to the directory holding the file I wish to edit or print and then invoke the batch file in my PATH directory. The batch file starts the proper program.

The Decision between PATH and Batch Files

As you recall, the PATH command tells DOS which directories should be searched to execute a program or batch file. If your programs are scattered across the hard disk, you must decide which programs should be invoked by a batch file in a PATH directory and which programs should be run simply by including the application's subdirectory in the PATH.

Again, the question has no right or wrong answer. A flexible guideline is that major programs with auxiliary files should be invoked through a batch file. If the program stands on its own, either place it in a PATH directory or create a new subdirectory and add it to the PATH.

For example, invoking 1-2-3 through a batch file makes sense. You need to be in 1-2-3's home directory because 1-2-3 expects certain auxiliary files in the current directory. Using a batch file for NewWord or WordStar also makes sense. These programs have many auxiliary files, which should be segregated into their own directories. The batch file for WordStar V3 does the needed setup to find the right directory holding auxiliary files or data files. The batch file for NewWord, which holds a single line, simply allows you to shorten your typing time.

Should you add the 1-2-3, WordStar, or NewWord directories to the PATH command? Probably not. Just put the application's batch file into a subdirectory that is included in a PATH statement (\UTILS or \BIN).

When you invoke a command, DOS examines the current directory, then the directories specified by PATH. If two directories are specified by PATH, as many as three directories are searched for any command (the current directory and the two named by PATH). When DOS searches directories listed by PATH, the search is made in the order in which the directories are specified by PATH. If the file is in the second directory, DOS searches only two directories. If the file is not in any of these directories, DOS searches three directories. Suppose that six or eight directories are specified by PATH. Misspelling a program name causes DOS to take a brief trip through many directories to find out that the program does not exist.

Even though the search time is brief, a good guideline is to restrict PATH to about five directories. The first directory in the list should hold the program-invoking batch files and your most frequently invoked programs. The next directory should have your next most frequently invoked programs, and so on.

The important distinction is *frequently run*, not *frequently used*. The CHKDSK command is frequently invoked, but you don't spend much time using the command. Many people invoke 1-2-3 once or twice a day but spend hours using the program. The object of the PATH command is to decrease the amount of time needed to invoke programs. Your greatest improvement comes when you speed the loading of frequently invoked programs, not the programs you use the most. However, don't take more than a few seconds to decide. The difference is only a hairline.

Another efficiency goal is to keep as few files as possible in the PATH directories themselves. The directory is a file, and DOS reads files in 512-byte sectors. Each entry in the directory is 32 bytes long, so a 512-byte sector holds the information of about 16 files. Longer directories (more entries) mean that DOS must wade through more entries to find the right file. Overzealous reductions in the number of files in PATH directories can be counterproductive, however; and the files that could be conveniently invoked through a batch file in a PATH directory should not themselves be in a PATH directory.

The general guidelines for deciding between the PATH command and batch files are

1. Use batch files to invoke major software programs, such as 1-2-3, rather than including these program files in a specified PATH.

2. Use the PATH facility to invoke major software groups, like the Norton Utilities.
3. When using PATH, give the directory holding the most frequently used programs to the command first, the less frequently used later.
4. Limit the number of directories given to PATH to approximately five. Keep nonessential files out of PATH directories.

Summary

When installing major programs, try to place the files in one directory; then sort the files into as many directories as logically needed. Try to keep data files separated from program files. If the program is uncooperative, you can use ASSIGN or SUBST to trick programs. Data path programs, which can be used for DOS pre-V3.1, also help (see Chapter 13).

Remember that when you are using subdirectories, the onus of hard-disk navigation is yours. Without direction, DOS does not know where the programs are located. You must explicitly inform DOS where these programs are located. You have three choices:

1. Build batch files that move to the correct directory and then invoke the program.
2. Build batch files that navigate DOS to the proper directory for the program but leave you in the current directory (do not switch the current directory).
3. Simply add the directory holding the program to the PATH command.

The best option depends on how well the program handles the hierarchical system with its program and data files. If you can easily change current directories within a program, make good use of the subdirectory system. Segregate files by purpose in different subdirectories. When the number of files in a subdirectory grows too large to manage comfortably, create new subdirectories and divide the files among the old and new directories.

If programs can run from any directory (have no auxiliary files) but do not allow path names with file names, move to the data file directory and invoke the program. If the program can run from only the current directory (uses auxiliary files), SUBST can trick the program into finding its own auxiliary files or into using separate subdirectories for data files.

This chapter also covers the following major points:

- Programs usually consist of several files: main program, overlays, auxiliary programs, and auxiliary data files. Installing and using a program can depend on how well a program knows where its files are located.
- Data files should be placed in separate directories. The number of different directories depends on how well and how conveniently the program uses subdirectories.
- The SUBST command allows you to use a disk drive name as an alias to a subdirectory name.
- The ASSIGN command reroutes activity intended for one disk drive to any other drive.
- You should limit the number of directories given to the PATH command to approximately five. Name the directories to PATH in the order of frequency of use. Keep your directories conveniently small.

This chapter concludes Part 2 of this book. Part 3 covers tips, techniques, and reviews of programs that make managing your hard disk easier.

Part **III**

Problem Solving

The three chapters of this part deal with different tasks and problems faced by the hard disk user. The chapters cover such issues as moving files between directories, locating information, recovering erased or damaged files and directories, and increasing system performance.

This part is organized differently from the remainder of the book. Because this part of the book serves as a reference, the chapters are organized by purpose. Operations involving files are covered in Chapter 10. Directory and disk operations are discussed in Chapter 11. Miscellaneous operations, such as reorganizing a disk, updating the operating system, and using RAM disks are explained in Chapter 12.

Although this part discusses several utilities provided with DOS, the text frequently refers to non-DOS programs. DOS simply does not have all the utility functions you need. For example, one common mistake is erasing the wrong file. DOS does not provide an easy-to-use program to recover from this mistake. An outside program is required.

I assume that you are willing to develop a tool kit of utility programs. These programs make the computer easier, more efficient, or faster to use. The goal of this section is to provide you with information about the right tool (program) for the task. You'll find that expanding your tool kit is amply rewarded.

Both commercial packages and shareware and public-domain programs are discussed. Most of the shareware and public-domain programs are available for a modest fee or no charge at all. These programs are invaluable. Appendix B contains a list of sources for the commercial programs cited, and gives more information about shareware and public-domain programs, and tells how to obtain them.

This part frequently gives the specific phrasing for commands, including any options (such as switches). I often cite the specific version used in the example. Remember that the only constant in the computer industry is change. As versions of the software change, new functions are added. Normally, the options shown in the discussion are correct, but more options will be added in later versions. Always examine the program's documentation to learn how to use any new features.

Also note that many functions performed by individual programs listed in this part also are performed by the hard disk managers and operating environments discussed in Chapters 13 and 14. After reading this part, turn to those chapters to review the capabilities

and features of the managers and environments. Your decisions about purchasing individual programs may be altered because you can use one of the managers or environments. You may find a single manager or environment that can replace dozens of individual programs.

I want to emphasize the conventions used in this book. For commands and programs, I use the following designations:

*dc:pathc***COMMAND** **required parameters** *optional parameters and switches*

The *dc:* is the disk drive holding the file to be executed. *pathc*\ is the path to the directory holding the program. The *dc:pathc*\ precedes each disk-based DOS command, program, and batch file. For DOS commands, the convention reminds you that the command is not built into DOS and must be loaded from the disk. Also remember that if the file is in the current directory or a directory specified in the PATH command, the *dc:pathc*\ may be omitted.

Also remember that items in *italic* are optional and are given as needed. Items in **bold** are required and must be given. Finally, items written in UPPERCASE (such as COMMAND) must be given exactly. Items in lowercase are variables, and you substitute the appropriate information for the variable (such as *a:* for *ds:*, the source disk drive name). Unless otherwise noted, DOS and all programs are not case-sensitive. You can normally type the command line in uppercase, lowercase, or both. A few commands are case-sensitive in regard to parameters or switches. These cases are noted in the discussion.

You will find the material in this part a valuable resource. After reading Part III, you can refer to the following Topical Index to find the different chapters and sections as you need them. Solutions to most of the problems you will encounter are given here.

Topical Index for Part III

Index by Program Name

Chapter 10

File Operations

In customary operations, users work more frequently on files than on directories. This chapter deals exclusively with operations on files, and the material covers a wide range of topics and programs. Many "gems" of information and advice are given in this and the next two chapters. I strongly suggest that you read the chapters through; then return later and reread them as your knowledge grows through experience. Some information that does not make sense on the first reading will become perfectly clear as you use your computer. File operations range from changing a file's attributes through copying and moving files to erasing files and recovering "lost" files.

Changing a File's DOS Attributes

DOS currently maintains six "bits" of information about each file in the directory. DOS, DOS programs, and outside programs use the different "bits" (attributes) for varied purposes. The seven DOS attributes are

archive
hidden
normal

read-only
subdirectory
system
volume label

Perhaps you're wondering how you get seven attributes from six bits. The seventh attribute is a normal file, which is simply a file without the hidden or system attributes turned on.

Calling the attributes "bits" is a meaningful pun. DOS stores the combined attributes for a file in one byte, and each attribute takes one bit of the byte. From least significant bit (right side or bit 0) to most significant, the order is read-only, hidden, system, volume label, subdirectory, and archive. Two bits are left for a future role in DOS.

These attributes are based on on-or-off binary information and are also called *flags*. To set the flag means to turn on the attribute; clearing the flag means turning off the attribute. Most computerists interchange freely the terms *attribute*, *flag*, and *bit*. For example, archive attribute, archive flag, and archive bit all have the same meaning.

Four attributes can be combined freely: read-only, hidden, system, and archive. For example, the operating system files (MIO.SYS and MSDOS.SYS or IBMBIO.COM and IBMDOS.COM) have the read-only, system, and hidden attributes set. When the files are copied to another disk, the archive bit is also set.

DOS automatically handles the setting of three attributes: archive, volume label, and subdirectory. When you are creating or changing a file, DOS automatically sets the archive flag. When you create or change a volume label, DOS sets this attribute on the entry. DOS flips on the subdirectory bit for the subdirectory's name (the name appearing in the parent directory) and the dot (.) and double dot (..) entries in the directories. The mechanism that sets the bits is an integral part of DOS.

Of the three automatic bits, only the archive bit also can be set and reset through DOS and outside programs. You can fiddle with the subdirectory and volume label bits with a disk-sector editor, but the result is seldom pleasing.

Of the three remaining attributes (read-only, hidden, and system), DOS provides ATTRIB to set and clear the archive and read-only flags. To clear and set the hidden and system attributes, you must use outside programs. The more flexible programs—SDACHMD of FilePaq and the File Attribute program of Norton Utilities—are discussed in the following sections. Some hints for manually setting the attributes to protect files and aid in program backup are also given.

Changing Attributes with ATTRIB

The ATTRIB command, added to DOS V3 and later, has two forms. The first form is for the ATTRIB command in DOS V3.0 and V3.1. The second form is for DOS V3.2. The difference is that ATTRIB in V3.2 also handles the archive bit; ATTRIB in V3.0 and V3.1 handles just the read-only attribute. The syntax shown is for DOS V3.2 and later:

*dc:pathc***ATTRIB** *sA sR d:path***filename**.*ext*

ATTRIB accepts a full file name (*d:path***filename**.*ext*). The file name designates which file(s) should have attributes altered or displayed. Wild cards are allowed in the file name.

ATTRIB accepts zero to two of the optional "switches." Unlike normal DOS switches, a switch character is not given. Instead, the *s* designates a *sign*, which precedes the attribute character to change. The form is + (set the attribute) or - (clear the attribute) immediately followed by an *A* (archive) or *R* (read-only).

The four combinations for the switches of ATTRIB are

+*A*	Set the archive flag (as if file has been created or changed)
-*A*	Clear the archive flag (as if the file has been backed up)
+*R*	Set the read-only flag (file cannot be changed or erased)
-*R*	Clear the read-only flag (file can be changed or erased)

Obviously, only one read-only and one archive switch may be given at a time. You may use one of the two sets (a read-only or an archive), one from each set (a read-only and an archive), or neither. If you use neither, ATTRIB displays the status of the read-only and archive attributes for the file(s).

To set the read-only flag of the file FINPLN02.WK1 and make the file unalterable and unerasable (assuming the file is in the current directory), you use the command

ATTRIB +R FINPLN02.WK1

To clear the read-only flag (make the file alterable and erasable), you issue the command

ATTRIB -R FINPLN02.WK1

To display the read-only attributes of FINPLN02.WK1, the command is

ATTRIB FINPLN02.WK1

ATTRIB places an *R* or *A* for the attribute at the left side of the screen if the respective attribute is set. If both attributes are set, the screen shows

```
RA      C:\123\REPORTS\FINPLN02.WK1
```

A blank appears where the letter normally is shown if the attribute is not set.

ATTRIB is a provided DOS program in DOS V3 and later and should be placed in the \DOS or \BIN directory. Also see the end of this section for tips on using the various DOS attributes.

Changing Attributes with SDACHMD

SDACHMD.COM, part of FilePaq, is the DOS attribute changer from SDA Associates. Although the name seems unfamiliar to DOS users, the name of a similar program, `chmod`, is very familiar to UNIX users. This program is used to *change* the permission-*mode* bits of a directory entry. In the permission-mode bits, UNIX records who may read, who may write, and who may execute a file. (UNIX also uses the bits to designate a subdirectory and show whether the file is actually a device.)

Just as the `chmod` program changes UNIX permission modes, SDACHMD changes the permission bit of DOS (read-only attribute) and the three other controllable attributes.

The SDACHMD syntax is similar to ATTRIB's but maintains more of the typical DOS style. (DOS's ATTRIB syntax is faithful to the UNIX style.) The syntax for SDACHMD is

*dc:pathc***SDACHMD** *d:path\filename.ext /P /T /N /As /Hs /Rs /Ss*

The *d:path\filename.ext* is the drive, path, and file name of the file(s) whose attributes are to be changed. Like the other FilePaq programs and unlike DOS, wild cards are accepted in both the file name and the path name. If no name is given, *.* is assumed.

The switches follow DOS's normal standards. They are presented in table 10.1. Note that in contrast to the DOS ATTRIB command, the plus and minus signs follow the letter of the switch. For example, suppose that you want to set all TXT files in the directory

C:\WORDS\LETTERS to read-only and clear the archive attribute. The command is

SDACHMD C:\WORDS\LETTERS*.TXT /R+ /A-

Table 10.1
SDACHMD Switches

Switch	*Function*
/P	*Pause* and confirm (query)
/T	*Traverse* the directory *tree*
/N	Change the file to *normal* (clear the hidden and system attributes)
/As	Set/clear the *archive* attribute
/Hs	Set/clear the *hidden* attribute
/Rs	Set/clear the *read-only* attribute
/Ss	Set/clear the *system* attribute
	s is a *sign* where + turns the attribute on - turns the attribute off

To reverse the settings (normal files, archive attribute on), you use the command

SDACHMD C:\WORDS\LETTERS*.TXT /R- /A+

To set to hidden all files in the 123 and associated subdirectories, use the command

SDACHMD C:\123*.* /H+ /T

The */T* switch traverses the directory tree. To reverse this setting, you use one of the following commands:

SDACHMD C:\123*.* /H- /T

or

SDACHMD C:\123*.* /N /T

These two commands are slightly different. The first command clears only the hidden attribute; the second command clears both the hidden and system attributes (normal files).

As with other FilePaq programs, an explicit or implicit wild-card file name causes SDACHMD to ask `Are you sure?` if the */P* is not used. If the */P* switch is used, the program skips the global query and asks for a separate confirmation for each file.

SDACHMD is more capable and flexible than ATTRIB. One other suggestion is to rename SDACHMD.COM to SCHMOD.COM or CHMOD.COM and place the program in the BIN or UTILS subdirectories. Other FilePaq programs are mentioned in the sections "Erasing Files" (SDADEL) and "Listing and Locating a File by Name" (SDADIR) in this chapter and in "Removing Directories" (SDARD) in Chapter 11.

Changing Attributes with File Attribute (FA)

The famous Norton Utilities, written by author-columnist-programmer Peter Norton, has so many excellent programs that every hard disk owner should buy a copy. File Attribute (FA.COM) is the Norton Utilities attribute changer. The syntax is similar but not identical to that of SDACHMD:

*dc:pathc***FA** *d:path\filename.ext /As /HIDs /Rs /SYSs /P /S /U /T*

Wild-card characters are accepted in the file name; and if no name is given, *.* is assumed. If you do not give any of the attribute-changing switches (*/As*, */HIDs*, */Rs*, or */SYSs*), FA displays the current attribute status of the files. The switches for this program are given in table 10.2.

Using the same examples as for SDACHMD, to set all TXT files in the directory C:\WORDS\LETTERS to read-only and clear the archive attribute, you use the command

FA C:\WORDS\LETTERS*.TXT /R+ /A-

To reverse the settings (normal files, archive attribute on), use the command

FA C:\WORDS\LETTERS*.TXT /R- /A+

To set all files within the 123 and associated subdirectories to hidden, the command is

FA C:\123*.* /HID+ /S

Table 10.2
File Attribute Switches

Switch	*Function*
/As	Set/clear the *archive* attribute
/HIDs	Set/clear the *hidden* attribute
/Rs	Set/clear the *read-only* attribute
/SYSs	Set/clear the *system* attribute
	s is a *sign* where
	+ turns the attribute on
	- turns the attribute off
/P	*Pause* when the display is full (as if MORE is used)
/S	Traverse the *subdirectories*
/U	Work only on *unusual* files (a file with any attribute set)
/T	*Total* only and do not display the individual files

The */S* switch (not the */T*) traverses the directory tree. The */T* switch silences the normal display of each affected file and shows only the total number of affected files. */U* restricts the search to files with archive, hidden, read-only, or system attributes turned on. Nonarchive, normal files without the archive bit on, are excluded.

File Attribute, and the entire Norton Utilities package, should be placed in \BIN, \UTILS, or a separate directory on the DOS PATH.

Using DOS Attributes

DOS has six possible attributes for a file; four are "user settable." The two attributes of wider interest are read-only and archive. Hidden and system are for experienced users and programmers.

Hidden and system files are not included in routine file searches, so the files are invisible to normal operations. These files are not displayed in the list of files (DIR) and cannot be copied or deleted. The hidden and system files are safe from accidental erasure and from "unknowledgeable" probing.

Hidden and system files can, however, be exposed by programs that expressly include these files, such as SDA Associates's SDADIR, Norton's File Find, and others. A hidden and system file is not

protected from deliberate change or erasure. Programs that explicitly call on DOS to open the file can alter the file's contents. DOS deletes the file when explicitly instructed to do so by a program, such as ZAP and SDADEL.

Although DOS treats these attributes identically, the two have different purposes. By convention, the system attribute is for DOS system files supplied by the DOS publisher. The hidden attribute is for the user's purposes. Files to be hidden by outside programs should use the hidden attribute. These conventions, however, are not binding and are often disregarded; some program publishers mark their invisible files as system and hidden.

The read-only and archive attributes are of more interest for your daily operation of the computer. The read-only attribute bars any alteration to or deletion of the file. Until the read-only flag has been cleared, the file is effectively frozen on the disk.

You'll note that the two DOS system files holding the BIOS and BDOS are marked as system, hidden, *and* read-only. The system and hidden attributes stop accidental alterations but do not stop explicit alterations. The read-only flag stops intentional alterations.

As mentioned, the archive attribute indicates that the file has been added to or changed since the last backup. The general use of the flag is to copy or back up selected files. Actually, the archive attribute indicates that a file has been created or changed since you ran the last program that cleared the archive flag. This slight difference is important because you also control the archive attribute.

Programs like BACKUP and XCOPY can select only files with the archive bit set. This restriction causes BACKUP and XCOPY to operate on new or modified files only. However, you can selectively alter which files should be included in such operations by manually changing the archive flag through programs like ATTRIB, SDACHMD, or FA. You have the ultimate control over the files that are selected by these archive attribute searchers. Because the archive attribute is used principally with backup, its use is further discussed in the Chapters 15 and 16, which deal with backup.

Generally, the system and hidden attributes are for more technical users. Feel free to experiment with the effects of these attributes, but do not leave the attributes in a tampered state without a compelling reason. If you have the proper change-mode (attribute) program, you can amaze your friends and coworkers by hiding and then using programs, batch files, or data files. If the hassle factor of using the machine is significantly increased, however, the problem you create for your coworkers and "soon to be former" friends outweighs the benefits.

If you have a logical reason to use the hidden attribute, *use the attribute's capability*. However, don't abuse the attribute. Follow established conventions and avoid setting the system attribute to hide files. The system attribute should be reserved for DOS system files. One compelling reason to change the hidden or system attributes is to remove damaged or out-of-date files, such as obsolete copy-protection schemes. If the copy-protected program's "uninstall" program fails or the copy-protection scheme's files are damaged, you may need to clear the system or hidden (and probably the read-only) attributes to erase the file. However, most extended delete programs (such as SDADEL) can erase these files if expressly directed.

The true puzzler comes when you issue DEL *.* on a directory and attempt to remove the directory with an RMDIR command, and DOS informs you that the path is invalid or the directory is not empty. A hidden or system file is the culprit. Use an extended DIR program to search for hidden or system files. If you find these files, use a change-mode program to clear the attributes and delete the files or use an extended delete program.

Certain disk foul-ups can leave a file's directory entry marked as a system or hidden file. Again, you need a change-mode program to restore the proper attributes to the file's directory entry.

The read-only attribute is a safe-haven attribute. Use the read-only attribute to protect files you do not want changed. Some beneficial uses are to protect all files in the root directory or all of a program's application files from accidental erasure. If you mark the files read-only, an inadvertent DEL *.* does not erase the protected files.

COMMAND.COM is the program for which the read-only attribute is most frequently set. Because many diskettes have COMMAND.COM (usually with an out-of-date version), the hard disk version is protected by setting the read-only attribute. When the files are copied to the root directory, the hard disk copy of COMMAND.COM is protected.

The read-only attribute is not a panacea. Data files, which are frequently changed, seldom benefit from this type of protection. By definition, these files must be alterable. You cannot freeze a program's auxiliary data files when the program, to function properly, must alter the file. Consistently marking and unmarking other data files as read-only is less than convenient. Do not substitute this sledgehammer approach (sporadically protecting files with the read-only attribute) for good computer operating procedures. Simply be careful when you erase or alter files. Make sure that the computer does what you want, not just what you type.

Avoiding Read-Only Traps

You can stumble into two traps when working with read-only files. When you copy a read-only file, DOS transfers the contents during a copy operation, but not the read-only, system, and hidden attributes. For example, when you copy a read-only version of COMMAND.COM to a floppy diskette, the diskette copy is not marked read-only. The diskette copy is naked and must be manually set to read-only.

The second trap is that the read-only attribute protects the file's contents but does not protect the file's name. A read-only file can be renamed. This unusual feature can produce unusual results, principally with some word-processing programs. Word processors usually create a new file for each revision of the document, and rename the old version using the extension BAK. If a BAK file for the document already exists, the older BAK is removed before the previous revision is renamed.

The calamity comes when you revise a read-only document the second time. The first revision causes the read-only document to be renamed as the BAK file. You have no problem yet. The BAK (the first generation of the document) is still protected, and the word processor successfully saves the first revision. When the second revision is to be saved, however, the word processor cannot delete the old BAK. Hence, the word processor cannot rename the first revision document to BAK, nor can the program successfully save the most current revisions. The editor's or author's astonishment (and subsequent anger) over the loss of the work (usually several attempts are made before the foul-up is discovered) is difficult to describe politely. Watch out for the side effect of successfully renaming read-only files.

Use the hidden and system attributes sparingly. Remember that system and hidden files are hidden from the uninformed only. Once the secret is obtained (you have the programs that expose the files or change the attributes), the secret is not a secret any longer. Use the read-only attribute when needed but do not substitute the read-only capability for vigilance in operations. Use the archive attribute as needed with backup or copy programs. More information on the archive attribute is provided in the chapters on backup.

Copying Files

A frequent chore for the computer user is copying files between directories, disk drives, or devices. The workhorse for these tasks is DOS's COPY command. You use COPY to copy one file or a set of files with related names. You also use this command to copy files between devices such as the keyboard, screen, printer, communications port, or the disk drives.

COPY falls short, however, when a single wild-card file name encompasses too few or too many files. Furthermore, COPY does not select files based on date and time stamps or DOS archive attributes, and the command does not copy hidden, system, or read-only files. As a result, the COPY command is ideally suited to handle about 80 percent of your copying tasks. To be effective in handling the other 20 percent, you need at least one different program. The choices include XCOPY, REPLACE (included with DOS V3.2 and later and discussed in Chapter 12), the shareware program PCOPY, and many of the programs mentioned in Chapter 13 ("The Hard Disk Managers") and Chapter 14 ("Operating Environments").

Copying files with the COPY command leaves the original files intact. A copying operation that erases the original set of files is called a *move*. File-moving programs are discussed in the section "Moving Files."

Copying Files with COPY

The COPY command is discussed as part of DOS Basics in Chapter 6. This section begins with a summary of that information and adds other facts about the COPY command.

The Syntax and Switches of COPY

The full syntax of COPY is

COPY */A/B ds:paths\filenames.exts /A/B dd:pathd\filenamed.extd /A/B/V*

The first file name (*ds:path\filenames.exts*) is the full name of the source file or files, the file(s) to be copied. The second full file name (*dd:pathd\filenamed.extd*) shows the destination of the copied file(s) and the new name(s). You can use wild cards in both the source and the destination file names.

As in most DOS commands, the name of a device can be used in place of a file specification. This fact is the underlying principle of the quick file-creation technique (COPY CON). Any device can be

used for the source or destination, but you get bizarre reactions from your computer if you copy from an output-only device (like a printer) or copy to an input-only device (like a keyboard).

Although all elements of the source name are shown as optional, this designation is not entirely true. The disk drive name is always optional, but you must include some part of a source file name. If you do not give a path name, you must give a file name. When you give only a path name in DOS V3, COPY assumes a file name of *.* and copies all files from the last directory in *paths*. In DOS V2, COPY always demands a file name.

All portions of the destination name are truly optional, as explained in Chapter 6. The "rule of currents" applies. Without a drive name, files are copied to the current disk. Without a path name, files are copied to the current directory of the disk drive. Without a file name, the copied file retains its original name. The only stipulation DOS enforces is that you cannot copy files to themselves. If you are copying files from the current directory or the current disk drive, you must specify one or more parts of the destination name, or COPY displays an error message and aborts the copying.

The COPY command has the following three switches:

/A Copy, assuming files contain *ASCII* text. Stop copying the source file at the end-of-file marker and add an end-of-file marker for the destination.

/B Copy, assuming files hold *binary* information. Copy the file according to the length recorded in the directory. Ignore end-of-file markers; do not insert any end-of-file marker.

/V *Verify* copy operation.

You are probably familiar with the */V* switch and the caveats about verify from Chapter 6. The switches new to you are */A* and */B*.

The */A* switch causes COPY to treat the file as ASCII text, which has a Ctrl-Z character (27 decimal, 1b hexadecimal) at the end of each file. (This convention is a hangover from the CP/M operating system, which did not track the exact size of a file.) The */A* switch is used primarily for copying a file to a nondisk device. COPY stops copying the source file at the end-of-file marker and places an end-of-file marker at the end of the destination file.

The */B* switch directs COPY to treat the file as binary information, in which a Ctrl-Z is a legitimate data character. The entire file is copied, based on the file's size given in the directory.

Because Ctrl-Z characters are treated as ordinary characters, a Ctrl-Z is not inserted in the destination file.

COPY assumes the */B* for all files, including text files. The entire text file is copied, including any Ctrl-Z. As a result, the end-of-file marker appears in the same place in the copied file as in the original file. Programs that depend on the end-of-file marker treat both the original and the copied files as the same.

The location of the COPY command's switches is unique compared to DOS commands. Switches can go before, between, or after file names. Because */A* and */B* switches are context-sensitive, a switch affects the immediately preceding file name and the files listed after the switch until another switch is given.

Table 10.3 shows the effects of placing switches in different positions in the command entry. As you can see in example 2, placing the */A* switch before all file names causes COPY to treat all the files as ASCII files. In example 3, the effect is the same as in example 2. Because a switch affects the preceding file and all following files, the */A* affects both files.

Table 10.3
Effects of the COPY Command's Switch Placement

Command	*File treated as ASCII*	*File treated as BINARY*
1. COPY SOURCE.TXT DESTIN.TXT		SOURCE.TXT DESTIN.TXT
2. COPY /**A** SOURCE.TXT DESTIN.TXT	SOURCE.TXT DESTIN.TXT	
3. COPY SOURCE.TXT /**A** DESTIN.TXT	SOURCE.TXT DESTIN.TXT	
4. COPY SOURCE.TXT DESTIN.TXT /**A**	DESTIN.TXT	SOURCE.TXT
5. COPY SOURCE.TXT /**B** DESTIN.TXT		SOURCE.TXT DESTIN.TXT
6. COPY SOURCE.TXT DESTIN.TXT /**B**		SOURCE.TXT DESTIN.TXT
7. COPY /**A** SOURCE.TXT DESTIN.TXT /**B**	SOURCE.TXT	DESTIN.TXT
8. COPY SOURCE.TXT /**A** DESTIN.TXT /**B**	SOURCE.TXT	DESTIN.TXT

In example 4, only the destination file, DESTIN.TXT, is treated as an ASCII file. The */A* switch given after the destination name does not affect COPY's handling of the source file. In this case, the entire contents of SOURCE.TXT are copied to DESTIN.TXT, and DOS adds an end-of-file marker to the end of DESTIN.TXT.

Examples 7 and 8 demonstrate contradicting switches. In example 7, the */A* switch treats the source file as an ASCII file. The */B* cancels the effect of the */A* on the destination file. SOURCE.TXT is therefore copied until the program reaches the first end-of-file marker, but no end-of-file marker is added to the destination file. In example 8, the last switch entered (*/B*) controls. The */A* switch affects only the preceding file (the source) because the */B* switch contradicts */A* for the destination file.

You can place a */V* (verify) switch before or after all file names. The */V* affects all files, regardless of the switch's position, because the */V* switch is not context-sensitive.

Another COPY Caution

Chapter 6 discusses some COPY command problems, particularly the DOS V2.1 COPY command, which develops amnesia every 256th file. A further caution, however, should be remembered.

Although careful not to copy files to themselves, COPY is absolutely "mindless" about overwriting files in different directories. When a source's file name matches an existing file in the destination, COPY erases the destination file and then copies the original source file to the destination file. The erasure is quiet; COPY does not display any message, so avoiding this disaster is difficult.

When replacing old files with new, you usually erase old copies before adding a new set of files. For this routine operation, you don't need to have COPY ask whether erasing the old file is acceptable. You want COPY to perform the task without intervention. The trap comes when two file names match, but the files themselves are different. You don't want the source file to replace the existing destination file. COPY, however, allows the second file to be destroyed.

If you are using the COPY command and are unsure of what files might be destroyed, test the source file name with the DIR command on the disk drive and path of the destination files. Check for matching names. Taking a few seconds to check prevents your spending needless time restoring data.

Copying Files with XCOPY

XCOPY (the DOS V3.2 and later *extended* COPY) is a hybrid of the COPY command and the BACKUP programs. The external XCOPY program copies files other than to or from a nondisk device. XCOPY's added features allow selective copying for making backups, keeping the files stored on several computers in sync (keeping all files the same), or manually selecting several files to copy rather than giving several COPY commands.

The syntax of XCOPY is

*dc:pathc***XCOPY** *ds:paths\filenames.exts dd:pathd\filenamed.extd /A /D /E /M /P /S /V /W*

The explanation of the source and destination file names is identical to that for the COPY command. You must specify at least the path (*paths*\) or file name (*filenames*) in the source. All other items in the source-file name and the entire destination-file name can be specified as needed.

XCOPY's power is reflected in the flexibility of its switches, which can be combined to include or exclude more or fewer selected files. You can look at the switches' functions in the order of complexity and then at some common uses of the XCOPY command. XCOPY's switches are listed and defined in table 10.4.

The */V* switch is identical to COPY's */V*; you *verify* that the copy is correct. Keep in mind that the cautions about verifying given in Chapter 6 also apply here.

The */P* switch is the *pause*-and-ask switch. The switch causes XCOPY to request confirmation before copying any file. The computerists' term for the "ask before you do" prompting is *query*. You use the */P* switch chiefly when a file name cannot include all the files you wish to copy or the file name includes some files you do not wish to copy. DOS displays the file name and requests a **Y** (yes) or **N** (no) response. If you type **Y**, DOS immediately copies the file.

Although the */P* switch lets you select individual files (a capability COPY does not have), I prefer other programs when I copy with a query. XCOPY forces me to wait while each file is copied before I can respond for the next file. I'm held captive to the program until it finishes copying each file individually. Because XCOPY does not accumulate "yes" responses before copying the files I want, I more often use a program called PCOPY, which I describe in the following section.

Table 10.4
XCOPY Switches

Switch	*Function*
/M	Copy files with archive bit turned on and created or *modified* since the last backup or XCOPY command with the archive bit turned on. XCOPY turns off the archive bit after the file is copied.
/S	Copy the source directory's and its subdirectories' files. This option is identical to the */S* of BACKUP, RESTORE, and other DOS programs.
/D:date	Copy files with date or time stamps equal to or later than the specified date. This option is identical to BACKUP's */D* option. The form of *date* is determined by the COUNTRY CONFIG.SYS command. For North American English-speaking users, the form is mm-dd-yy (month, day, year).
/A	Copy files with the *archive* bit on (like the */M* switch) but does not clear (turn off) the archive bit after the file is copied.
/E	Used with */S*, */E* creates parallel subdirectories on the destination directory even if the directory is *empty*.
/P	*Prompt* and ask for confirmation before copying each file
/V	*Verify* that data is written to the disk correctly (equivalent to turning on COPY /V or VERIFY ON).
/W	After XCOPY starts, the program prompts and then *waits* for you to insert any disks necessary to search for designated files.

The */W* switch tells XCOPY to *wait* so that you can insert needed diskettes. XCOPY then loads the data from the diskette into memory. As soon as the data is entered, the screen displays a message telling you to press a key to begin the search-and-copy process.

The */A* and */M* switches select files with the DOS archive bit turned on. DOS turns on the archive bit when a file is created or modified. Certain programs, such as BACKUP, automatically turn off the bit when processing the file. The only difference between the two switches is that */M* clears the archive bit after copying the file,

but */A* does not. A discussion of the proper switches to use in different applications appears in Chapter 15.

The */S* and */E* switches handle the way XCOPY uses directories. The */S* switch affects the source and destination directories; the */E* switch affects only the destination.

The */S* switch tells XCOPY to work with files in a specified directory and all its subdirectories. Although COPY limits itself to files in one subdirectory, XCOPY can traverse the directory tree. As an example, look at a model directory from Appendix A. The following command copies all files from the \123\FINPLAN directory on drive C to the current directory on drive A:

XCOPY C:\123\FINPLAN*.* A:

The following command copies from the various subdirectories of 123 all worksheet files with names starting with F:

XCOPY C:\123\F*.WK1 A: /S

The copied files, however, are not placed in the current directory of drive A. XCOPY creates a parallel directory structure on drive A before copying the files to the appropriate directories. As an example, imagine cutting the directory tree off at 123, shaking the tree until all unwanted files (files that do not start with an F and end in WK1) and unwanted subdirectories (subdirectories of 123 that do not have the F*.WK1 files) fall off, and then transplanting a copy of the tree in the current directory of A.

The actual process is as follows:

1. XCOPY examines 123 for files matching the name F*.WK1. No files are found.
2. XCOPY then searches each subdirectory of 123. The FONTS and GRAPHS subdirectories are skipped because they hold no qualifying files. FINPLAN, however, has such files.
3. XCOPY creates the FINPLAN subdirectory in the current directory of drive A and then copies the qualifying files (FINPLN02.WK1, FINPLN03.WK1, and FINPLN11.WK1) into the new FINPLAN directory.
4. SALES also contains a qualifying file. XCOPY creates a new SALES directory in the current directory of A and then copies the FORECAST.WK1 file to the new SALES directory.

If FONTS, GRAPHS, FINPLAN, or SALES has subdirectories, XCOPY examines these subdirectories for qualifying files. The search continues in the subdirectories until the end of the directory tree is reached.

The */S* switch traverses directories and establishes parallel directories at the destination. If a path is specified in the destination, the new directories start in the specified destination directory. For example, the preceding command can be changed to

XCOPY C:\123\F*.WK1 A:\123 /S

This command places the SALES and REPORT directories in A:\123. Similarly, the following command established a 123 directory on drive A:

XCOPY C:\F*.WK1 A: /S

Although neither 123 nor the remaining directory structure on drive C has a matching file, the \SALES and \REPORT directories must be established in a parallel directory. Therefore, XCOPY creates a 123 directory on drive A before creating the \SALES and \REPORT directories. Because no files in the C:\123 directory match the F*.WK1 name, the SALES and REPORT subdirectories are the only entries XCOPY places in the A:\123 directory.

The */E* switch causes XCOPY to create empty subdirectories. During the search for F*.WK1, XCOPY finds no matching files in FONTS or GRAPH. Suppose that the following command is given:

XCOPY C:\123\F*.WK1 A: /S/E

XCOPY must establish a parallel set of subdirectories on the destination, even if no files are copied to those subdirectories. This command creates the \SALES, \REPORT, \FONT, and \GRAPHS subdirectories on drive A, although \FONT and \GRAPHS are empty. No qualifying files are copied into these two directories.

With two major differences, XCOPY's capabilities closely resemble those of the DOS BACKUP command. BACKUP splits files over more than one diskette. After the backup copy fills the first diskette, BACKUP takes care of storing the file on additional diskettes. XCOPY, however, halts when the destination disk fills. You must change diskettes and reissue the XCOPY command (use the */P* switch so that files already copied can be skipped). The other difference is that files copied with XCOPY are not altered and are directly usable. Files backed up with the BACKUP command must be restored before the files can be used. The XCOPY command is discussed again in Chapter 15.

Copying Files with PCOPY

PCOPY.COM is a shareware program written by Norm Patriquin. PCOPY is not a superset of COPY but, rather, is a superior COPY program. The program manipulates files with a plethora of options and is usable on any version of DOS V2 and later. Another Patriquin program, PMOVE.COM (a file-move program), uses a nearly identical syntax.

The syntax for PCOPY is

*dc:pathc***PCOPY** *ds:paths***filename**.*extdd:pathd**filenamed.extd /switches*

PCOPY's source and destination file-naming conventions follow those of COPY. If you fail to give either a source or a destination file name, PCOPY displays a help screen.

The */switches* are PCOPY's switches, which determine the files to be copied and other actions. The switches listed in table 10.5 demonstrate only a portion of the full capability of the program. Unlike most DOS programs, some switches start with the same letter (such as the */D-* and */DC* switches) but have different meanings.

Table 10.5
PCOPY Switches

Switch	*Function*
/A	Select files with the *archive* bit set
/B	Select files with the archive *bit* (like /A) set. Clears the archive bit on the original file after the file is copied.
/C	*Copy* only (used with the PMOVE program)
/D–	*Delete* files on the target directory before copying
/DC	*Copy* the *directory* and the file. The file is copied to the same directory on the target disk. If a target directory does not exist, it is created.
/D:sdate	Select files based on their *date*. The switch's syntax is */D:*, followed by an optional selection character (*s*), followed by a mandatory *date*. The selection characters are (all files older than *date*) all files younger (newer) than *date* = all files with the same *date*

Table 10.5 (cont.)
PCOPY Switches

Switch	*Function*
	If a selection character is not given, a newer than date symbol is assumed. The date can be a calendar date or number of days (a positive whole number). The syntax for a calendar date can be one of the following: mm/dd/yy mm-dd-yy ddmmmyy *dd* represents a two-digit day, *yy* a two-digit year, *mm* a two-digit month (such as 10/4/87 or 10-04-87), and *mmm* a three-letter month (such as 10OCT87).
/E	Select only files that *exist* on both the source and destination so that duplicate files on the destination can be replaced
/F:name	Indicate the *first* file. This switch selects the named file (*name*) and all files appearing in the directory after that *name*. Files with names appearing before the file *name* are omitted.
/L:name	Indicate the *last* file to select. This switch omits all files in the directory that occur after *name*.
/M	Pause the display after each screenful (like the program *MORE*)
/ME	*Merge* two directories. All files that do not exist in the destination are moved (copied and erased) from the source. The most recent copy of a duplicate file is retained. The switch deletes on the source any remaining duplicate files that are not copied because the target directory holds a more current version. This switch is identical to giving the */U* and */X* switches together.
/N	Move from the source files that do not exist in the destination (*new* files)
/NW	Direct the program not to use windows (*no windows*). PCOPY normally uses windows.
/O	*Omit* files specified by the source file name. This switch effectively reverses the source file

Table 10.5 (conc.)
PCOPY Switches

Switch	*Function*
	specification. All files that match the source file name are omitted, leaving the files that do not match the source file.
/P	*Pause* and ask (a query)
/R	Unconditionally *replace* files on the target disk regardless of date
/RE	*Register* the shareware copy of the program
/RO	Include files marked with the DOS *read-only* attribute
/S	Include the entire directory *system* in the search. Unlike typical DOS use, the */S* switch causes PCOPY to begin at the root directory and include all subdirectories given. To traverse subdirectories that start with the current or specified source directory, use the */DC* switch.
/SA	*Save* existing files on the destination. Rather than overwriting duplicate files on the target directory, PCOPY saves the files by renaming them. The last two characters of the extension are changed to a number from 00 to 99. Duplicated files are numbered in ascending order.
/SF	Include files marked with the DOS *system file* attribute
/T	*Test* the operation. PCOPY displays the result of the intended operation without performing the operation.
/U	Select source files that do not exist on the target directory, or source files that are newer than matching files on the target directory (*updates*)
/V	*Verify* the copy operation (identical to COPY's */ switch)*
/X	Move (copy and erase) the selected files

Because many of PCOPY's features work like features of COPY and XCOPY, a full description of all the PCOPY uses is not given.

Instead, the following paragraphs explain some of PCOPY's unique operations.

With the *date* switch, you can specify files dated on, before, or after a specific date or number of days. The following command copies from the current directory all files created or modified *after* April 10, 1987:

PCOPY *.* A:\SAVE /D:)4-10-87

The following command copies files created or modified *before* April 10, 1987:

PCOPY *.* A:\SAVE /D:(4/10/87

Note that files created or modified *on* April 10, 1987, are not included in either operation. To copy these files, you use the = character.

To operate on files created or modified in the last 10 days, the command is

PCOPY *.* A:\SAVE /D:)10

To operate on files created or modified more than 10 days ago, the command is

PCOPY *.* A:\SAVE /D:(10

Another useful pair of switches is */F:name* and */L:name*. One problem frequently faced by COPY users is that the target diskette is filled and the COPY command aborts. You can instruct PCOPY to start where COPY stopped by using the */F:name* switch. Use the name of the file that COPY was copying when the diskette filled. PCOPY then copies the named file and all files that appear in the directory list after the named file. */L:name* also limits the files to be copied. No file whose name appears after the named file is included. The */F:* and */L:* switches can be used to copy a range of files from a directory.

The */O* switch inverts PCOPY's selection. Instead of including all files that match the source file name, PCOPY selects all files with names that do *not* match the source file name. Note that unless the */S* (entire system) or */DC* switch is given, PCOPY inverts the selection for only the file name portion of the full file name. PCOPY does not invert the disk drive or the path names.

/T is the *test* switch. If you are unsure of what PCOPY will do, you can include a */T* switch. PCOPY indicates what would happen to the files but actually does nothing. You can safely test file names and the effects of various PCOPY switches without touching the files.

PCOPY is one of three shareware programs by Patriquin. PMOVE is a windowed move (copy/delete) utility. PSEARCH is a windowed search utility that combines the capability to search a file's contents with the capability to search for files by name. Many people believe these three utilities are worth their weight in gold. To use these programs to the greatest advantage, place them in the \BIN or \UTILS subdirectory. Use COPY when one file name takes care of everything you want to do. PCOPY, however, is an excellent program to use when your copying needs are more complex.

Destroying a File

As discussed in the section "Recovering Files," a deleted file is not literally removed. Although a deleted file's directory entry and disk sectors are marked as reusable, the old information remains intact until new information overwrites the file.

If you need to destroy beyond recovery the confidential contents of a file, the WIPEFILE.COM program from Norton Utilities performs well and is dangerous only if you ignore its warnings. Use WIPEFILE.COM with the utmost caution. Even though WIPEFILE has several built-in safeguards, you should employ common sense when using the program. Once WIPEFILE has done its work, the file *cannot* be recovered. The information is obliterated.

The syntax for the command to invoke WIPEFILE is

*dc:pathc***WIPEFILE** *d:path\filename.ext /Vn /Rn /G /NOD /N /P /LOG*

d:path\filename.ext is the full file name of the files to destroy. Wild cards are accepted in the file name, but I strongly suggest that you avoid them. Give the exact file name to prevent accidental destruction of files you don't want to lose. The switches for WIPEFILE are described in table 10.6.

WIPEFILE automatically pauses before beginning its operation. If the */P* or */N* switch is given, WIPEFILE asks for your confirmation before operating on each file.

To destroy all LET files in the current directory, for example, use the command

WIPEFILE *.LET

To conform to governmental standards for file destruction for all files in the C:\WORDS\LETTERS subdirectory and also confirm before destroying each file, give the command

WIPEFILE C:\WORDS\LETTERS*.* /G /P

Table 10.6
WIPEFILE Switches

Switch	*Function*
/Vn	*Value* that WIPEFILE will use to overwrite the old information. *n* is a value from 0 to 255.
/Rn	Number of times WIPEFILE is to *repeat* the overwriting operation
/G	Follow *governmental* rules for secure file wiping
/NOD	Do *not delete* files after wiping information
/N	Do *not* wipe file, simply erase the file (*/P* switch is implied)
/P	*Pause* and ask before wiping each file (query)
/LOG	Format the output for redirecting to a printer or disk file *log*

To help prevent the program from being invoked accidentally, the file name WIPEFILE.COM should not be abbreviated. WIPEFILE.COM may be placed in \BIN or \UTILS or in a PATH directory that holds other Norton Utilities files. If better security from accidental invocation is desired, the program should be placed in a separate directory that is not on the DOS PATH.

Erasing Files

Removing unwanted files is a common housekeeping chore in disk management. The most common commands for this task are the DOS ERASE or DEL (delete) commands, which are built into the command processor.

Although ERASE and DEL are useful commands, more sophisticated programs for removing files are available. The SDADEL program, a part of SDA Associates' FilePaq, is a good example of a more efficient approach to removing files. Both DOS's ERASE and DEL and SDADEL are explained in this section.

Removing a File with ERASE or DEL

ERASE and DEL are cousins. The syntaxes for these commands are

ERASE *d:path\filename.ext*

or

DEL *d:path\filename.ext*

The name of the file you want to remove is *d:path\filename.ext*. You must enter a path name, a file name, or both; wild cards are accepted in the file name. If you give only a path name, ERASE/DEL assumes a file name of *.*.

When you use a wild-card file name, the command asks for confirmation that all files in the directory are to be deleted. The following prompt appears:

```
Are you sure? (Y/N)_
```

The prompt is displayed only when all files in a directory will be erased. If you type **Y** and press Enter, ERASE/DEL proceeds with the deletions. If you type **N** and press Enter, the command is stopped.

Using a wild-card file name that takes in too many files is the ERASE/DEL command's most common problem. A file you intended to keep can disappear inadvertently. To avoid an unpleasant surprise, use DIR and the intended wild-card name to preview the lists of files you plan to erase.

See the section "Recovering Files" for information about recovering erased files.

Removing a File with SDADEL

SDADEL.COM, a superset of the DOS ERASE/DEL command, is a program included in SDA Associates' FilePaq. The program can erase hidden or read-only files, query the user before removing each file, and hunt files in the directory system. (The other parts of the FilePaq package are SDADIR, SDACHMD, and SDARD. SDADIR and SDACHMD are mentioned elsewhere in this chapter; SDARD is mentioned in Chapter 11.)

The syntax for SDADEL is

*dc:pathc***SDADEL** *d:path\filename.ext /T . . . /T /P/H*

SDADEL accepts a full file name. Wild cards may be used in both the file name and the path name. SDADEL and the FilePaq programs are

among the few programs that allow a wild card in the path name. If no file name is given, *.* is assumed.

SDADEL has three switches:

/T *traverses* the subdirectory tree (SDA Associates calls this the *tree* switch, similar to the DOS TREE program)

/P *pauses* and asks before each file is erased. The switch needs to be included only once in the command line.

/H deletes *hidden* files as well. Include this switch only once in the command line.

The traverse switch (*/T*) affects only the full file name preceding the switch. If you want to search the directory tree for more than one file, you must add a */T* switch after each full file name.

When you use the */P* switch, a prompt appears that shows the file name and asks whether the file should be deleted. You respond with a *Y* (yes) or *N* (no) answer. If you have used a wild-card file name (or implied a wild card by giving no file name) without including a */P* switch, SDADEL asks `Are you sure (Y/N)` and waits for a *Y* or *N* answer.

The DOS ERASE and DEL commands do not "see" hidden-attribute files and cannot erase these files. The */H* switch directs SDADEL to delete hidden files. SDADEL also deletes files marked with the DOS read-only attribute. SDADEL acts as if the */P* switch is set and asks for confirmation before each read-only file is deleted.

I use the /P switch with SDADEL more than the other switches. The */P* switch lets you give a wild-card file name, and then select individual files to delete.

The second most frequent use of SDADEL is to remove word-processing backup files from the disk with the */T* switch. Many word processors automatically create backup files for each revision of a document. These files typically have a BAK extension. The following command hunts down and deletes all BAK files from the current disk:

SDADEL *.BAK /T

Most users prefer to use the built-in DEL command for removing files when a single file name includes all necessary files in a directory but no unwanted files. SDADEL is better when you need a query capability or want to traverse the directory tree.

Here's another general tip: Before you begin using the SDADEL program, you may rename SDADEL.COM to SDEL.COM and place it in the BIN or UTILS directory. SDEL is much easier to remember.

Listing, Locating, and Displaying Files

Locating a file, either by name or by contents, is a frequent problem when you use the hierarchical directory system.

One way to prevent the problem of "losing" a file is to use meaningful file names—root names and extensions that follow a systematic pattern and give an indication of the file's contents. This accomplishment is not easy, given the limit of 11 user-definable characters you are permitted for file names, but the systematic approach can be rewarding. The process is easier for extensions than for root names; for example using the LET extension for files that hold letters. Meaningful root names, such as FINPLAN7 and MARSALES (March sales), are more difficult to construct, but the 8 characters can convey considerable information.

Even the most experienced user, however, may "lose" a file among the myriad directories on most disks. The problem occurs more frequently than one would guess. Perhaps the user remembers the file name (or parts of the file name) but not which directory holds the file. Or the user remembers that some file holds a specific line or group of characters, but can't remember the name of the encompassing file.

The solution to both problems is covered in this section. Because the more frequent problem is locating the file by name, that problem is covered first. The section on SDADIR also covers how a file takes more room than you think.

Listing and Locating a File by Name

Finding an errant file can be painless. Like the solution to many other problems, this solution depends on the quality of your tools. You can use five different programs or tools: DIR, TREE, WHEREIS, SDADIR, and FF. DIR and TREE are provided with DOS, but the other three programs are better tools for the task. One is a public domain program; the second and third are parts of utility sets. This section starts with the DOS-provided programs and then covers the outside programs.

Locating a File with DIR

The DOS DIR command is suited to finding a known or partially known file name within one or several directories. DIR also shows other vital information about the file's date and time of creation or last modification and the file's size. Because DIR is an internal command, invoking the command causes no delay. The syntax is

DIR *d:path\filename.ext /P/W*

All arguments to DIR are optional. A missing disk drive or path name causes DIR to use the current disk or current path of the disk. A missing file name is interpreted as *.*, or every file in the directory. Wild cards are allowed in the file name (*filename.ext*) but not for the drive or path name.

The */P* switch causes DIR to *pause* when a screenful of information is displayed. Pressing any key signals DIR to continue the display. The */W* switch causes DIR to display file names in the *wide* (five-across) style, omitting the date/time stamp and size statistics for each file.

When you search for a known file name, use the specific name. When you search for a file with a partially remembered name, use a wild card that *probably* matches the file. Use as few literal (not wild-card) characters as possible to increase the chance that the file will show in the list. If the file name could be FILE4.LET or FINAL4.LET, the most probable name for DIR to use is FI*.LET or F*.LET. If the extension is unknown, use FI*.* or F*.*.

Similarly, give as few literal characters as possible when searching in one directory for a set of files with related names. When several of the files are located, use a broader or narrower wild-card name to find the remaining files.

I/O redirection is useful with DIR. The listing can be directed to a disk file for editing and later printing or to the printer for an immediate hard copy.

DIR also displays the names of subdirectories located in the directory you are viewing. DIR does not display hidden or system files.

Locating a File with TREE

The DOS TREE command, which is used principally to show the hierarchy of the directories on the hard disk, also can locate a specific file. Normally, TREE prints the names of all directories on

the disk. The optional switch forces TREE to print the names of the files in each subdirectory. The syntax of TREE is

*dc:pathc***TREE** *d: /F*

Because TREE is a disk-based external command, you use the optional disk drive and path name for the command. TREE accepts a single optional parameter and the disk drive name. Be sure to give the disk drive name if a drive other than the current one should be searched.

To locate a file, the */F* (*file*) switch must be used. The name of each normal file (without hidden or system attributes turned on) is displayed under the appropriate directory name.

Unfortunately, TREE is not selective. You get either no file names (when you omit the */F* switch) or the names of all files on the disk. Because you cannot specify a branch of the system, you get the entire directory hierarchy. The output of TREE is too long for locating one or two files.

The best use of TREE is to provide a road map of files and directories for other users of the system. As with the DIR command, you can redirect the output of TREE to a disk file or to a printer.

Locating a File with WHEREIS

WHEREIS.COM is a public-domain program designed to locate a file anywhere in the subdirectory system. The name is appropriate. Unlike DIR, WHEREIS automatically traverses the entire directory structure to locate a given file. The general syntax of WHEREIS is

*dc:pathc***WHEREIS filename**.*ext*

WHEREIS accepts a single file name without a path. Wild cards can be used in either the mandatory root name or in the optional extension. For example, to find the file HANDY.EXE the command is

WHEREIS HANDY.EXE

WHEREIS prints the full path of each occurrence of HANDY.EXE on the disk. To locate all BASIC files, the command is

WHEREIS *.BAS

Because WHEREIS traverses the entire directory structure, use as specific a file name as possible. Otherwise, WHEREIS displays too many files and takes longer than needed.

Often, you may wish to stop WHEREIS with Ctrl-Break as soon as the file you want is displayed. Because WHEREIS does much disk

work and little display, DOS ignores the Ctrl-Break request until WHEREIS displays another file. If WHEREIS takes longer than you like, set BREAK ON, run WHEREIS, and press Ctrl-Break after WHEREIS has displayed the desired file. Then set BREAK OFF. If you use this procedure frequently, place the commands in a batch file.

You will find that WHEREIS is a highly useful and frequently used program. For greater convenience, I recommend that you place WHEREIS in a PATH directory such as UTILS or BIN. If you wish a shorter name, change WHEREIS.COM to WHERE.COM.

Locating a File with SDADIR

SDADIR.COM is part of SDA Associate's FilePaq. A superset of the DOS DIR command, SDADIR optionally can traverse the subdirectory tree to find files. Other optional switches give extensive control over what files or directories to view and the type of output.

The syntax of SDADIR is

*dc:pathc***SDADIR** *d:path\filename.ext/T . . . /T /P/W/5/2/H/B/D/DO/FP /S=*

The exceptional nature of SDADIR starts with its treatment of file names. *d:path\filename.ext* is the drive, path, and name of the file to display. If you omit the file name, SDADIR assumes that you want every file in the directory. You can, however, give more than one full file name to SDADIR; simply use a space to separate full file names (designated by the ...). Wild cards are accepted in *both* the path and the file name. SDADIR is one of the few programs that accepts wild cards for a path name.

Before the discussion of SDADIR's abundant switches, look at Figure 10.1 for an example of the output of SDADIR. Notice that the program shows the file name (columns 1 and 2), and date and time stamps (columns 6 and 7) as does DIR. The name of the displayed directory is also shown. However, SDADIR shows much more. The third column lists the DOS attributes for each file. If the following letters appear in this column, the appropriate attribute is turned on for that file:

A	*archive* file
H	*hidden* file
S	*system* file
R	*read-only* file

Command line: **SDADIR C:\ /H**

```
SDADIRectory - V3.00 - (C) Copyright 1984 SDA Associates -
Run on  9-22-1986

Files Found In Directory: C:\

Name     Ext  Att.     Used/Allocated   Date     Time

IBMBIO   COM  ASHR    16369/    16384  7-25-86  4:31p
IBMDOS   COM  ASHR    28477/    28672 12-30-85 12:00p
CML0203  HCL   SHR     9344/    10240  8-11-86  1:24p
COMMAND  COM          23791/    24576 12-30-85 12:00p
CONFIG   SYS             39/     2048 12-23-85  9:28a
CONVDI   SYS            114/     2048  6-12-86  2:23p
FP       CNF             11/     2048  5-29-85  3:14p
OPTIMIZE EXE  A H     18432/    18432  8-11-86  1:25p
START    BAT  A         831/     2048  8-21-86  6:39a
STARTS   BAT  A         469/     2048  5-29-86  3:47p
VDF0203  VDW   SHR     1296/     2048  8-11-86  1:25p
VDI      DRV  A         194/     2048  3-18-86 12:00p
     99367/    112640 bytes used by 12 files
    714752 bytes remaining
```

Fig. 10.1. *Output of SDADIR.*

Note that the first three entries show combinations of these attributes. The two DOS system files show all attributes turned on. The third file, CML0203.HCL, is a hidden Prolock file used in the copy-protection process.

The fourth and fifth columns show the actual space used by the file and the amount of space DOS has allocated to the file. As explained in Chapter 2, DOS allocates space by clusters, not disk sectors. Although a file may be one byte long, DOS allocates a full cluster to the file. When the file grows past the full space of the cluster, additional clusters are allocated to the file, one at a time.

On a single-sided HC (high-capacity, 1.2M) floppy diskette, a cluster is 1 sector, or 512 bytes large. On double-sided diskettes, each cluster is 2 sectors or a total of 1,024 bytes per cluster. On 10M hard disks, each cluster is 8 sectors or 4,096 bytes. Most 20M and 30M hard disks use 4 sectors per cluster, or 2,048 bytes. Depending on the medium, a file holding a single byte of information

is allocated 512, 1,024, 2,048, or 4,096 bytes of disk storage. As the file grows, it fills the unneeded space and, upon crossing the cluster boundaries, receives more space.

DIR shows the actual space the file is using. SDADIR also shows in the `Allocated` column how much space DOS has given exclusively to the file. As you have guessed, your disk storage is never 100 percent efficient. You can waste up to 30 percent of your storage capacity with files that "hog" more disk space than the files actually use. This waste is a natural consequence of using DOS. If you use files, you waste some disk space. The worst case is using many short document files on a 10M hard disk, where each file is less than 1K in length and DOS allocates 4K to each file. However, the information in Chapter 12 on archival programs can help you throw many infrequently used short files into a large common file.

A bytes used/bytes allocated total is given for each directory. If you have used the */T* (traverse) switch, a used/allocated total is shown for all directories displayed.

SDADIR switches. SDADIR's 11 switches are listed and defined in table 10.7. The */P*, */W*, */5*, and */2* switches control the display of files. The */P* (pause) and */W* (wide) switch serve the same purpose as DIR's switches. SDADIR adds a synonym for the */W* switch, a */5* switch. When the 5-across file display is used, only the file names are displayed. The */2* switch allows a display two names across. The */2* display also includes the file's actual size, date, and time but omits the file's attributes and allocated size.

The */H*, */B*, */D*, and */DO* switches control which files are displayed. The */H* switch includes hidden files, which are normally omitted in a directory listing. Figure 10.1 was produced using the command **sdadir \/H**. The */B* switch shows the files that have the archive attribute turned on. The */B* can show which files would be used if the commands **BACKUP /M** or **XCOPY /M** (or **XCOPY /A**) were given. The */D* switch instructs SDADIR to include directories in the list. Although DIR shows subdirectories, SDADIR must be invoked with the */D* switch to display subdirectories with the files. The */DO* switch instructs the program to omit files and show only subdirectories.

The */T* and */FP* switches control the subdirectories that are searched. The */T* switch, used to traverse the subdirectory tree, searches the starting directory and all subdirectories. The */T* switch affects the preceding file name only. If you give more than one file name and you want SDADIR to traverse the directory tree for each name, a */T* must follow each full file name.

Table 10.7
SDADIR Switches

Switch	*Function*
/P	*pause* after a screenful
/W	*wide* (five-across) display of file names
/5	*5*-wide display (identical to */W*)
/2	*2*-wide display
/H	Show *Hidden* files also
/B	Show files whose archive attribute is set (the file which would be *backed* up if the command BACKUP */M* were given
/D	Show *subDirectories* also
/DO	Show *subDirectories Only*
/T	*Traverse* the directory system (identical to the */T* switch of many DOS programs)
/FP	Show the files in the directories covered by SDA Associates' *FilePath* program
/S=	*Sort* the output based on one or more of the following characteristics: N file *name* E file *extension* S file *size* D file *date* A file *attributes* T file *time*

The */FP* switch directs SDADIR to show those subdirectories covered by FilePath, a data-file PATH program by SDA Associates, which is discussed in the section "Data Paths" in Chapter 12. */T* and */FP* cannot be used together.

The */S=* option *sorts* the output. The switch has two forms: */S* sorts the list by file name and extension in ascending order. */S=* accepts several keys and allows sorting in ascending or descending order. The form of the */S=* switch is

/S=k+/-

The *k* is the single character *key* (N, E, S, D, A, or T, listed in the switches) and is immediately followed by either the plus sign (+) for sorting in ascending order or the minus sign (-) for sorting in descending order. To give several keys, enter the keys one after the other without spaces between. For example, to sort the output by ascending order on the file's name and the extension, use the switch

/S=N+E+

This switch is the same as giving */S*. The command

SDADIR *.WK1 /T/D-N+

shows a list of WK1 files in the root directory sorted by descending date (the newest files first) and ascending name (normal alphabetical order). By including the */T* switch, you search the entire disk for WK1 files.

Tips for Using SDADIR. Popular uses of SDADIR are displaying hidden files using the */H* switch, finding all occurrences of a file using the */T* switch, showing the subdirectories of a directory using the */DO* switch, and showing the entire file and directory structure of the disk using both the */T* and */D* switch.

Remember that SDADIR starts in the current directory if a path name is not given. Omitting the root directory path name is a common mistake users make when searching for all occurrences of a file (using the */T* switch). If you are searching the entire disk for a file called MYFILE.TXT, give the name \MYFILE.TXT. If you do not give the path name, SDADIR starts in the current directory and continues the search through the subdirectories tree of the current directory. In other words, SDADIR searches only part of the disk.

As with DIR, the output of SDADIR can be redirected to a disk file or to the printer. Although SDADIR is "noisier" (displays more information) than the TREE /V command, SDADIR is more comprehensive; it is preferred for printing a subdirectory map of the entire disk.

If you wish to shorten the name, I suggest changing SDADIR.COM to SDIR.COM. The file should be placed in the UTILS or BIN subdirectory. You should be aware that a public-domain program named SDIR.COM exists. If you copy that program onto your hard disk, you could overwrite the renamed SDADIR. You would, however, have little use for both the public-domain SDIR and SDADIR.COM.

My personal preference is to use WHEREIS.COM for locating files on the entire disk. I use SDADIR for its expanded directory listing and for selective directory searches to preview names for the FilePaq

erase program (SDADEL). I also run SDADIR on the entire hard disk periodically to check the efficiency of my storage. When free space gets low, I look through a printout from SDADIR for the files I can move from the hard disk or simply erase.

I highly recommend FilePaq, which contains SDADIR and other programs, even though some functions are duplicated by WHEREIS and by the next program discussed—the Norton Utilities FileFind.

Locating a File with FF (FileFind)

FF.COM, or FILEFIND.COM, is a lesser-used member of the Norton Utilities, but it accomplishes the task of locating files. Because I prefer shorter to longer file names, I use FF instead of the more meaningful but longer name FILEFIND.

Two of the three switches are similar to DIR's switches:

*dc:pathc***FF** *d1:filename.ext d2: d3: . . . /P/W/A*

FF accepts an optional file name. If no file name is given, *.* is assumed. The disk drive names are discussed with the switches.

Two of the three switches are similar to DIR's switches:

/P	*pause* when a screenful is displayed
/W	*wide* (five-across) display
/A	search for the file on *all* disk drives

FF has the unique ability to search more than one disk drive for a file. A disk drive name (*d1:*) can be part of the file name. Disk drive names also can be included after the file name (*d2:*, *d3:*, and other names designated by the ...). The disk drive names should be separated by spaces. The shortest method to search all disk drives is to use the */A* switch.

FF precedes each list of matching files with the name of the directory containing the files. The program also displays the file's size (used, not allocated, size), the time in 12-hour, am/pm format, and the date in full form (day-of-week name, month-of-year name).

This program, and the entire Norton Utilities, can be placed in the \UTILS or \BIN subdirectory. My preference is to place the utilities in their own directory, called \NORTON, and specify this directory in the PATH statement.

Locating and Displaying a File by Content

This section covers two topics: locating a file by its contents and displaying a file's contents. The two functions are similar because both "wade through" the contents of a file.

An occasion when you must play detective is when you try to locate a file based on its contents. Usually, you don't remember the file's name, but you do know a specific phrase or line in the file. This kind of detective work is done frequently by users of word-processing programs or by programmers who create and maintain scores of files and must locate a specific file after a lapse of time.

Using the hierarchical directory system helps narrow the possible locations of sets of file names that hold the desired information. Good file-naming conventions also help. However, at times a file-by-file search is necessary.

The type of search is based on the file's contents: you have a wide range of choices for ASCII text files, but a limited range of choices for non-ASCII files.

When you search an ASCII text file, what you see is what you get. The characters or numbers in the file are identical to what is on the screen, so you easily can display the file's contents on the video display or printer. Some program tools are helpful, the DOS TYPE and MORE commands, for example. A non-DOS MORE program, the TS (TextSearch) program, and the GREP program also help locate files.

With non-ASCII files, what you see on the display is not what is in the file. Often the information, particularly numeric information, is compressed or altered. Frequently, you do not know what to search for because you do not know how the information is represented in the file.

Non-ASCII files include program files and some data files. Searching data files can be especially difficult. Users of data files frequently are looking for specific numeric information that may have undergone an obscure transformation. Sometimes the only way to locate a data file is to invoke the program used to create and maintain the file, such as 1-2-3 or Framework for spreadsheets or an accounting program for a receivables file. Searching data files one-by-one takes time but is often the only method available.

Happily, most programs do not compress textual (character-oriented) information. Most programs store the strings as ASCII text in a data file. Therefore, you have a reasonable "shot" at searching for ASCII information in these data files.

Remember, however, that certain word processors may use data that "looks like" ASCII text but is not. WordStar and its clone, NewWord, are examples. WordStar sets the most significant bit for the last character in each word and for nonparagraph carriage returns. When you use the TYPE command on a WordStar file, you see some recognizable characters and some "nonsense." The nonsense consists of the characters with the most significant bits set. When you search though these files, you may miss the string for which you are searching because some characters are altered by the word processor.

Displaying a File's Contents with TYPE

The first command in the content-search/display repertoire is DOS's TYPE command. TYPE is built into DOS and is always accessible at the DOS command line. The syntax of TYPE is

TYPE *d:pathc***filename**.*ext*

The disk drive, path name, and extension are optional. A root name for a file is required. TYPE accepts wild-card names, but only the first file with a name matching the wild-card name is displayed. TYPE works with only a single file at a time.

TYPE shows the contents of a file on the video display as ASCII text, which means that program instructions or packed data appear as nonsense. The output of TYPE can be redirected to a file or printer. This procedure is generally worthless, however, because the file can be copied to another file or printed using PRINT or a similar command.

The output of TYPE (and any other DOS command) can be suspended by the Ctrl-NumLock or Ctrl-S key sequence. Ctrl-NumLock is preferred over Ctrl-S because if any character is accidentally pressed before Ctrl-S, DOS does not "see" the Ctrl-S character and does not freeze the display. Ctrl-NumLock always works. To free the screen, press any key. To break out of TYPE, use the Ctrl-Break or Ctrl-C sequence.

As mentioned, TYPE works with a single file at a time. To use TYPE with a wild-card file name, use the following one-line batch file, which I call TYPER, and place the file in a PATH directory:

```
FOR %%D IN (%1 %2 %3 %4 %5) DO TYPE %%D
```

The FOR..IN..DO batch file command allows you to use more than one file name and expands wild-card file names into individual names. Because null batch file parameters are ignored, one to five full

file names, which include wild cards in the file name, can be given. When the correct file is located, press Ctrl-Break or Ctrl-C to end the batch file.

Often you will see enough of the file's contents to know that you are searching the wrong file. If you are using a single TYPE command, you can simply press Ctrl-Break to get out and then issue the command again. Pressing Ctrl-Break also aborts the batch file; you do not have a graceful exit when running the batch file.

Displaying a File's Contents with MORE

DOS's MORE.COM program is a simple filter with a simple purpose. The program (discussed in Chapter 5) displays a screenful of information and the prompt `--MORE--`. Because the program looks for standard input (from the keyboard), you can easily redirect input to MORE.

To type a file one screenful at a time, use

*dc:pathc***MORE** <*d:path***filename**.*ext*

Because MORE is an external program, you must direct DOS to its location or keep MORE in a PATH directory (preferably DOS or BIN). Use the less-than symbol (<) to redirect MORE's input from the keyboard to a disk file.

The advantage of this technique is that you see a screenful of information at one time. You can read the file's contents at your leisure without frantically trying to press Ctrl-Break before something scrolls off the top of the display. The disadvantage is that you cannot skip quickly to the needed information. If you suspect that the information you are looking for is at the end of the file, you end up using seemingly endless keystrokes to page through the display. In this case, the TYPE command is preferred.

Because DOS's MORE program does not accept file names, you must use DOS's I/O redirection to get a file's contents to MORE. DOS also does not allow wild-card names with its I/O redirection. However, you can construct a MORER batch file to use a wild-card file name:

```
FOR %%D IN (%1 %2 %3 %4 %5) DO MORE < %%D
```

This batch file has the same advantages and disadvantages as the TYPER batch file. If you use the file, place it in a PATH subdirectory.

Locating and Displaying a File by Content with a UNIX-like MORE

MORE's roots are in the UNIX operating system. `more` is part of the University of California at Berkeley's implementation of UNIX—formally named *bsd* (Berkeley Software Distribution) and nicknamed Berkeley UNIX. (UNIX programs and commands use lowercase letters in their names. I follow the appropriate convention for each operating system and use the digital typeface so that the UNIX names are distinguishable.)

UNIX distinguishes between upper- and lowercase letters; DOS changes names to uppercase letters. (Most industry observers humorously attribute the UNIX lowercase convention to programmers who cannot locate the Caps Lock key on their terminals.)

DOS patterns its I/O redirection after UNIX. While DOS's I/O redirection is fairly faithful to UNIX, DOS's MORE does not capture the convenience or power of the bsd `more`.

The MORE programs accept file names, so you need not redirect input to MORE. These programs also accept more than one file name on the command line and accept wild cards in the file names. Therefore, a batch file is not needed. You can search several files with one command. With a MORE program, you can move the file's display one line or one screenful at a time. Pressing the space bar tells MORE to advance by one page; pressing the Enter key instructs MORE to advance by one line. This feature is useful when you want only the next few lines in the file while keeping some lines on the current screen.

Some MORE programs offer other features, such as showing the percentage of the file that has been displayed, skipping forward or back a specified number of lines, or skipping to a line containing a string specified by the user. The public-domain MORE programs are preferred for searching the entire contents of several files. These programs are more useful than DOS's MORE and, while not an absolute necessity, the programs are too useful to shun.

If you choose to get the MORE program, remember DOS's search order on the PATH. If you place your public-domain MORE file in UTILS, DOS retrieves first the MORE that is in DOS. You must relocate DOS's MORE to another directory or erase it from the disk to prevent DOS's retrieving the wrong file. If you place MORE in BIN, some confusion is still possible. Most public-domain MORE programs are EXE files. DOS's MORE.COM is run before a MORE.EXE. Again, move MORE.COM or eliminate it.

Locating and Displaying a File's Contents with FIND

The TYPE and MORE programs are based on linear searches that show the entire file's contents on the video display. The problem is that the display of the file takes too much time. That approach works best when you must display the entire file or the first part of a file.

The approach used by the next group of programs is to search a file for a specific word, line, or phrase. The computer does the difficult work of searching and comparing and then displays the lines containing the information you desire. You need not display the whole file to confirm that the information you want is in the file.

The first of many content-search programs discussed is FIND, which is provided with DOS. FIND searches a file for a specified string. The output of FIND can show you the lines in the file that hold the string, the lines that do not contain the string, or just a count of the lines that have the string.

The syntax for the FIND.EXE program is

*dc:pathc***FIND** */V/C/N* "**string**" *d:path\filename.ext* . . .

The phrasing of FIND is different from that of most DOS commands. First, the switches for FIND are given between the word FIND and the string, which is set off by quotation marks. The switch position conforms to the UNIX style. In normal DOS style the switches are given after all other information. The switches are

/V	display all lines that do not have *string* in them
/C	*count* the lines that have *string* in them, but don't display the lines
/N	*number* each line as it is displayed. The number shown is based on the line's position in the file, independent of the occurrences of *string*.

The *string* is the group of characters for which FIND searches. The string can be almost any length and *must* be enclosed by double quotation marks. As you may expect, spelling counts. What you may not expect is that capitalization counts. FIND is one of the few commands where upper- and lowercase characters are viewed differently. *MEMORY* is not the same as *Memory* or *memory*. FIND's matching is literal.

You can specify more than one file for FIND to search. The file names must be separated by spaces and must conform to the conventions of disk, path, and file names. FIND does not accept wild cards in the file name. You can, however, overcome the wild-card

problem with the same type of batch file as you used with TYPE or MORE. If you do not specify any file names, FIND looks for input from the keyboard (standard input).

For example, to find and number each line containing the phrase *memory chips* in the files CHAPT1.TXT, CHAPT2.TXT, and APPEND.LET, the command is

FIND /N "memory chips" CHAPT1.TXT CHAPT2.TXT APPEND.LET

FIND displays the name of the file and the lines that hold the string (no */V* switch) or the lines that do not hold the string (*/V* switch). Or FIND gives the count of the lines (*/C* switch). If you are looking through several files and anticipate many "hits" (matches), redirect the output to a printer, a disk file, or to the MORE command. Because FIND is literal, giving a shorter string and having too many matches is better than giving too long a string and taking the chance that FIND will skip the desired line.

Locating a File's Contents with TS (TextSearch)

TS.COM is the TextSearch program of the Norton Utilities. Like the next program (GREP), TS is a superset of FIND. Although overshadowed by the Norton "unerase" program, TS is powerful and deserves attention.

You can run TS several ways. The syntax to search for a string in one or more files is

*dc:pathc***TS** *d:path\filename.ext "string" /S /T /LOG /N /EUR /EBCDIC*

The syntax to search the disk and ignore file boundaries is

*dc:pathc***TS** *"string"* **/D** or **/E** */LOG /N /EUR /EBCDIC*

When you issue the command in the first form, you can leave out every item on the command line except **TS**. If you do not give a file name, TS asks you for a name. If you do not give a string, TS asks for the string.

Only one full file name may be given, but wild cards are allowed. If the string is included on the command line and is longer than one word, the string must be enclosed in double quotation marks.

TS is insensitive to the case of characters. You may use upper- or lowercase letters in the *string* and, unlike FIND, TS matches either case.

Either or both of the first two switches, */S* or */T*, can be given. The meanings of these switches are

/S	traverse the *subdirectories* during the search
/T	enter *totals* mode

The */S* switch is identical to the subdirectory switch of other DOS commands. The */T* switch sets the totals mode, meaning that the names of files holding the string are displayed without a pause. When you don't give the */T* switch, TS pauses, displays the section of the file where the match is found, and asks whether the program should search for another match.

In the second form, TS searches the disk, not individual files. One of the two switches in boldface must be given:

/D	search the entire *disk* for the string
/E	search the whole disk but only in areas not currently used by other files (*erased* areas)

The */D* switch unconditionally searches the entire disk for the string. The */E* switch checks only areas not in use by files. The */E* switch allows you to search areas where erased files or abandoned portions of a file may exist.

The remaining switches can be used with either form of TS:

/LOG	reformats TS's output for redirection to a printer or *log* file
/N	adapts the output for *non-IBM* computers
/EUR	handles *European* characters (ASCII values > 127)
/EBCDIC	handles data using the *EBCDIC* character set

Suppose that you are using TS to search through a set of files to find to whom in California you've written letters. The command line may look like this:

TS \WORDS\LETTERS "CA"

Because TS is not case-sensitive, the *CA* matches *California* or the abbreviation *CA*. However, the string also matches any other words that begin with *CA*, including *can*, *Canada*, and *capable*. If you suspect that too many words begin with CA, use the */T* flag to get a list of files that match the string and then examine the files individually.

The quietest output from TS comes when you use the */T* and */LOG* switches. The names of the files holding the strings are displayed with a summary of the number of matches.

A minor shortcoming of TS is that it does not handle "funny" files—data files that look almost like ASCII files. WordStar files are an example. When I used TS to search for the word *switches*, TS did not find a match. Because I know that WordStar clips the last character of the word, I shortened the search word to *switche*, and TS found the words.

TS is a valuable program that is included in the Norton Utilities. See the section on FileFind for recommendations on its placement.

Locating and Displaying a File's Contents with GREP

Another UNIX legacy is the super FIND program `grep`, whose name is an acronym for something like *g*et *r*egular-*e*xpression *p*attern or *g*lobal *r*egular-*e*xpression *p*rint. Several public-domain and commercial versions of `grep` are in the DOS world. Their "faithfulness" to the full-featured UNIX `grep` varies. In some cases, DOS GREP programs are nothing more than the FIND program with wild-card file names allowed. Other versions capture UNIX features.

A major feature of GREP is its capability to search for ranges of characters. The specified string is interpreted by GREP, and any string that matches the interpretation is displayed. Because the string is not matched literally, you can think of the string specified to GREP as a *pattern*, or a model.

The GREP discussed here comes from the Text Management Tools sold by Lattice, Inc. Except for minor differences, Lattice's GREP fully implements the features of the UNIX `grep`. A complete discussion of GREP takes more space than this book allows; this section presents the most important information and shows several examples.

The syntax of GREP is

*dc:pathc***GREP** *-p -n -c -f -v -s -V -$* **pattern** *d:path***filename**.*ext* . . .

Because GREP has many UNIX-like deviations from DOS, this discussion covers the standard portions of command first and then the exceptions.

GREP allows you to specify one or more file names on the command line, and the file names may contain wild-card characters. File names are the last part of the command line.

GREP uses the UNIX form for switches. The switches occur after the command name, GREP, but before other information. The switch letters are case-sensitive. *V* and *v* designate two different options. UNIX uses the minus sign (-) instead of the slash (/) as the switch character. As with other UNIX programs, you have two choices when you give multiple switches. You may place the minus sign in front of each character and separate the switches with spaces, or you may use a single minus sign and group all the switches into a single word. For example, to use the *p* and *v* switch you can write either

-p -v

or

-pv

As in most programs, the order of the switches does not matter. The meanings of the switches are given in table 10.8.

Table 10.8
GREP Switches

Switch	*Function*
-V	Print the *version* number of the program
-n	Turn off line *numbering* (normally the line number is printed before each line)
-f	Print only the names of the *files* where the match was found
-c	Print only a *count* of the match lines
-v	Print only lines that do not contain the *pattern*
-p	Filter non*printable* characters (useful for WordStar or similar text files)
-$	Ignore case distinctions (upper- and lowercase are viewed as the same)

The *-n*, *-c*, and *-v* switches are identical to switches in DOS's FIND. The *-f* switch allows you to suppress the printing of lines and get just a list of the files that hold the pattern. *-p* overcomes the problem with data files that look almost like ASCII files (discussed in the section on TextSearch). *-$* allows you to control whether uppercase letters should be viewed differently from lowercase letters.

The *pattern* is the string to search. The string must be enclosed in double quotation marks. *pattern* can contain normal characters, which will be matched literally, or the metacharacters or sequences listed in tables 10.9 and 10.10.

Table 10.9
GREP Metacharacters

Character	*Function*
\	GREP's escape character (not an ASCII Escape character) prefixes escape sequences
.	Matches any one character (similar to the ? in file names) but does not match a newline or end-of-line ($)
^	The beginning of a line
$	The end of a line
[	Beginning of a group
]	End of a group
!	Not one of the characters in a group (only meaningful when the ! is the first character after the [). UNIX uses the circumflex (^) instead.
*	Matches zero or more occurrences of the preceding character or group
+	Matches one or more occurrences of the preceding character group
-	Range character (only meaningful when used within a group []).

Table 10.10
Escape Sequences

Character	*Meaning*
\b	Backspace
\n	Newline (carriage return-line feed)
\r	Carriage return
\s	Space
\t	Tab
\xnn	The ASCII character designated by the hexadecimal (base 16) number nn
\c	The actual character represented by c. The \ is used to cancel the special meaning for the of c, which may be ! $ ^ * - + [ ] . \ "

A simple example of a pattern that uses no special characters is

grep "user guide" articles.txt

This command finds the lines containing the words *user guide* in the file ARTICLES.TXT. To find lines that contain the words ending in *he*, you use this command:

grep ".he" articles.txt

The period matches any single character. **".he"** matches the words *she*, *the*, or *ahe*. This pattern also matches *he* in the document because the period matches any character, including numbers, spaces, or punctuation characters.

When you need to match the period character itself, you must temporarily cancel its special meaning to GREP. To overcome GREP's special meaning of the period, you prefix each occurrence with a backslash. The \. translates to a period. Similarly, the "\$" translates to a $. To find the amount $1,234.56, you use

grep "\$1,234\.56" articles.txt

Remember that the backslash is called an escape character but is not the ASCII Escape character (CHR$(27) or (0x1b); the \ tells GREP, "Pay attention; this is for you."

Character groups or *classes* are a major power of GREP. Classes are used to match a possible range and a single character, meaning the class matches only one character. For example, to search for any occurrence of *the* regardless of case, you use the pattern:

"[Tt]he"

Each character group is enclosed by brackets ([]). The **[tT]** becomes the letter *t* and then the letter *T*. GREP looks for *the* and then *The*. Similarly, you can look for the words *the* and *she* by using **"[st]he"**. To search for any possible capitalization of *the* and *she*, you use multiple groups:

"[STst][Hh][Ee]"

Each set of groups reduces to a single character as GREP spins through the pattern looking for upper- and lowercase character combinations.

Suppose that you are searching for dates in your file. You can use character classes to find any year in the twentieth century:

"19[0123456789][0123456789]"

This pattern matches the digits 19, a single digit (0–9), and another single digit (0–9). Because typing the entire set of numbers is time-consuming, GREP allows the following type of notation for a range of characters:

low character-high character

For example, **[0–9]** matches any number from 0 to 9. You can replace your earlier pattern with

"19[0–9][0–9]"

A shorthand method to match a single letter and number (omitting punctuation symbols) is

"[A–Za–z0–9]"

This pattern specifies the letters *A* through *Z*, *a* through *z*, and *0* through *9*. When you give a range, be sure that the first character has an ASCII value lower than the second character. *A* has an ASCII value of 65; *Z* has an ASCII value of 90. Writing **[Z–A]** is a mistake.

You can also use classes to look for characters not specified in the class. The **!** is used for this purpose and must directly precede the opening bracket. The class **[!1]** matches any character except the

number 1. Similarly, the class **[!A-Z]** matches any character not in the range A-Z. The shorthand method to match any nonalphanumeric character is a simple flip of the preceding example:

[!A-Za-z0-9]

(UNIX users will note that Lattice uses ! where UNIX programs use the ^ for the negation character.)

The ! has special meaning only when it is the first character in a class. In the case of **[0-9#!]**, the ! is interpreted as a ! rather than as the Not operator.

The ^ and $, called *anchors*, fix the match to the beginning or end of a line. When you know that the pattern you seek is at the beginning or end of a line, these characters help you eliminate spurious matches. To search a document for all lines that end in a period, you use

grep "\.$" articles.txt

To search for sections in a document that start with a number, a period, a tab, and an alphabetic character, you use

grep "^[0-9]*\.\t[A-Za-z]" articles.txt

The meaning of this line is "Start at the beginning of a line (^) looking for a number from 0 to 9. Match as many numbers as you can (the combination of the group [0-9] and the *). The following character should be a period. After the period, the next character should be a tab character. After the tab should be a letter, either upper- or lowercase. If this is true, you have a match."

To find the lines ending in a closing quotation mark, you use

grep "\"$" articles.txt

Inside the customary opening double quotation mark, you first enter another double quotation mark (which must be escaped or GREP thinks you have ended the pattern). The $ indicates that nothing should follow your search pattern. Only lines ending in a double quotation mark will be shown.

Have you ever needed to find a control character in a file? FIND and TS do not handle this type of search, but GREP excels. If the control character is not a "newline" (carriage-return-line-feed combination, ASCII values 13 decimal-0d hexadecimal/10 decimal-0a hexadecimal), backspace (8/8), tab (9/9), or a space (32/20), you must give the corresponding ASCII value in hexadecimal form. For

example, Ctrl-S (WordStar's underline code) has an ASCII value of 19 decimal or 13 hexadecimal. To search for any lines with a Ctrl-S, you use the pattern:

"\x13"

The boldface character for WordStar is Ctrl-B, 2 decimal and hexadecimal (numbers under 10 are the same in both counting systems).

I know that I want my headings in boldface and that the boldface characters are the first and last characters of the line. I also know that my headings do not exceed one line. To search for my headings, I use this command:

grep "^\x02.*\x02$" myfile.txt

The ^ and $ anchor the beginning and end of the line. The **\x02** designates the Ctrl-B character. The **.*** matches any characters in between.

The GREP discussed here suitably represents the public-domain and commercial versions of GREP. Lattice's Text Management Utilities (TMU) includes a file comparer (DIFF), a word counter (WC), and a UNIX-like stream editor (ED) and related utilities. TMU also includes a file mover (FILES).

When you get a GREP program, I suggest that you place it in your UTILS or BIN directory. Because GREP is used mainly with text and word processors, you might make a subdirectory called \WORDS\BIN. Place GREP and similar tools in this directory and specify the directory in the PATH command.

GREP is a versatile search program that requires experimentation to master, but once mastered is indispensable.

Renaming a File

A common chore in file maintenance is changing a file's name. DOS builds this function into the command processor; the function is always available at the command line. The syntax is

RENAME *do:patho***filenameo**.*exto* **filenamen**.*extn*

The RENAME command accepts a modified phrasing, with wild cards acceptable in both file names. The first name is the current file's name. I add an *o*, for old file name. This name is a full file name. The new name for your file is designated with the *n*. The new name is

simply a file name, not a full file name. The name cannot contain a disk drive or path name. Because you are changing the file's name, not altering the file's location, DOS considers the other information extraneous.

DOS's refusal to accept duplicate names prevents RENAME from giving two files the same name. The system displays the error message

```
Duplicate file name or File not found
```

This message indicates that RENAME has stopped because a new file name would duplicate an existing name.

To replace an old file with a new file with the same name, you must use the COPY command. When copying files, you can preserve an older file by changing its name. The older file's name then does not match the copied file's name. One frequently used technique is to change the extension and keep the root name the same. Examples of this technique are changing DAT (data files) to DTA or changing the extension to OLD.

Moving Files with MV

Astonishingly, DOS does does not provide the capability to *move* files—copy files to a new location and then delete the originals. The need to move files occurs when you are reorganizing directories or transporting files from one machine to another and wish to eliminate the possibility of using out-of-date files.

Several public-domain, shareware, and commercial programs for moving files are available. Some programs move just files; only PCTools moves complete directory trees (see Chapter 13). This section covers moving files from one subdirectory or disk to another.

One public-domain program that moves files is MV.EXE. This program is based on the UNIX program `mv`. Although `mv` moves complete directory trees, MV.EXE handles only files. The command for MV has two possible forms:

*dc:pathc***MV** *ds:paths***filenames**.*exts* *dd:pathd***filenamed**.*extd*

or

*dc:pathc***MV** *–v* *ds:paths**filenames.exts* . . . *dd:***pathd**

You can use wild cards in the file names with either form. The first form illustrates a file move that can rename the file(s) during the

move. This command is equivalent to using COPY to copy and rename a file(s) and then using ERASE to remove the original file(s). The second form shows a move without changing file names. The two forms are significantly different.

In the first form, you must give two file names. In the second form, you may give one or more different source file names, following the pattern for full file names. The last name given is the destination, which cannot contain a file name. The destination name must be the directory to which the files will be moved.

For example, to move all sample BAS files that come with DOS from \BIN to \BASIC\SAMPLES, the command is

MV \BIN*.BAS \BASIC\SAMPLES

To move the two BASIC program files (BASIC.COM and BASICA.COM) at the same time you move the \BAS files, the command is

MV \BIN\BASIC?.COM \BIN*.BAS \BASIC\SAMPLES

MV.EXE should be placed in the \BIN or \UTILS directory.

Recovering Files

Damaged or corrupted files are the bane of any computer user. DOS handles the first problem satisfactorily, and outside programs can cope with both situations. This section discusses these common problems and presents possible solutions.

Damaged and Corrupted Files

Damaged and corrupted files are not identical. A damaged file has one or more bad sectors. DOS indicates that the file cannot reliably be read or written at one or more spots on the disk. A corrupted file, on the other hand, can reliably be read or written, but the information inside the file is incorrect.

Files are damaged or corrupted most often by electronic and mechanical malfunctions. Electronic malfunctions are the most frequent and produce several disastrous results. A runaway program (or program responding to incorrect operator input) writes the wrong information or writes the correct information into the wrong file.

Faulty memory or a faulty program corrupt the in-memory copy of DOS, which in turn erratically records the file or key sections of the disk (the file allocation table or directory). A sudden power loss prevents DOS from completing a file-write operation, or the disk electronics malfunction. Disk electronic failures are the most frequent cause of damaged files; errant programs are frequently the cause of corrupted files.

Mechanical failures (infrequent for hard disks, more frequent on floppy disk drives) include head crash, which damages the disk's media; disk ball-bearing failures, which cause the platters to wobble so that the disk cannot reliably read information; and the physical wearing out of a floppy diskette.

Regardless of the cause, your concern is to restore the file. As you recall from previous chapters, files are classified into programs, their auxiliary files, and data files. Basically, programs and their auxiliary files should not be restored. The damaged or corrupt program file should be eliminated from the disk and a copy of the file restored from the original program diskette.

Missing or damaged portions of a program file produce disastrous effects. At best, a faulty program file causes the machine to lock up. At worst, the program appears to work but is damaging data files on the disk. Generally, you should not bother to "mend" program and auxiliary files.

Damaged data files can be recovered, but some information (in multiples of 512 bytes, the size of a disk sector) probably will be lost. A partially recovered damaged data file is like a corrupted data file.

The success of recovering a corrupted file depends on the type of information in the file and what information is lost. Word-processing files are usually the easiest to restore. You simply edit the document and add missing information or delete useless information. ASCII data files are the next easiest to restore. You use a text editor to correct or add information.

Non-ASCII data files are the most difficult to restore. Deciphering the information inside the file is difficult, and knowing how to phrase the missing information may be impossible. Generally, to restore a data file, you must write a special program or use a disk-sector editor (a program that allows you to manipulate information on a disk on a sector-by-sector basis rather than a file-by-file basis).

Using the DOS RECOVER Program

The DOS RECOVER program has two forms: a file mender and a disk mender. The disk-mending function of RECOVER is explained in the section "Recovering Directories and Disks" in Chapter 11. The disk mender has the potential to turn useful files into undecipherable nonsense and should be used with extreme caution. For this reason, *always* give a file name to RECOVER when you are attempting to recover a single file.

RECOVER attempts to read as much of the file as possible and places the contents in a new file with the same name. Information that cannot be read is skipped. One good side effect of RECOVER is that the bad sectors of the disk are marked as in use so that these sectors will not be used again. The syntax for the file-recovering form of the command is

*dc:pathc***RECOVER** *d:path***filename**.*ext*

The *d:path*\ is the optional disk drive or path name of the file. If the file name has an extension, be sure that you include both the root name and the extension. Wild cards are accepted in the file name, but RECOVER operates on the first file with a name matching the wild-card name. Sometimes you may not reconstruct the file you want.

Using RECOVER on an undamaged file is harmless; you waste only the time DOS spends reading and writing the file. Using RECOVER on damaged files is beneficial because the bad sectors holding the file are "removed" from use. Although the information in the bad sectors may be lost, some information is reconstructed from the bad file.

RECOVER does not truly remove bad sectors; the sectors are simply marked as in use by a nonexistent file. For this reason, RECOVER can be a decent "quick and dirty" method to hide bad sectors. To remove bad sectors, you must reformat the disk. For floppy diskettes, use the FORMAT command. FORMAT may work for a hard disk, but sometimes a physical format program is needed (see Chapter 2). Because FORMAT or the hard disk physical format program wipes out all existing information, you should back up your disk first.

You may find that bad sectors mysteriously "appear" on your hard disk. Many people who purchased the early IBM ATs with CMI disk drives experienced higher than average troubles. In my own case, over the course of a year's time, the AT I used gradually developed

"flaky" sectors. As I added more information, my hard disk would write to an area previously unused and another couple of questionable sectors would appear.

At times, the AT would grind away once or twice and then go on its merry way. (You learn quickly to recognize the sounds the disk makes when it encounters a bad sector.) In those cases, the hard disk wrote the sector with marginal success. In other cases, the machine would grind away until I received the disturbing `Disk read error on drive C` or a similar message. When asked to `Abort, Retry, or Ignore?` I would press **R** for retry; and the machine could sometimes read the sectors successfully on the second or third attempt. (DOS makes several attempts to read marginal sectors.) Sometimes, though, the retry option would not work and DOS consistently reported failure.

For that case, I would either erase the file and re-create it or use the DOS RECOVER command and manually reconstruct the lost information. For the first two cases, I would manually copy the file over to a floppy diskette, erase the original, and use a utility (like Norton Utilities' DiskTest program) to mark the bad sectors permanently.

If many bad sectors appear, you should examine the entire hard disk for problems. Programs like DiskTest, the Mace Utilities REMEDY, and Kolod's HTEST are discussed in "Recovering Directories and Disks" in Chapter 11.

Restoring Corrupted Files

Before attempting to salvage a corrupt file, *always* make a copy. If you alter the file beyond hope, you have a copy of the file available to try again. If you suspect a serious problem with the disk, place the copy on another disk. Keep the copy intact until you are sure the file has been successfully mended.

A corrupt file must be edited. If the file is ASCII text, you can use a text editor or word processor in programmer's or nondocument mode to delete or add the needed information. The procedure can be complicated when an unwanted end-of-file marker traps wanted text. As you recall, Ctrl-Z is the customary end-of-file (EOF) marker. Most text editors stop reading information when they encounter the EOF marker. If a Ctrl-Z appears in the file before the end of your text, information is trapped and will not be read.

If the file is small (can be held in memory), you can use DOS's DEBUG to fix the file. If the file is larger, you must use a disk-sector

editor such as the NU program from Norton Utilities to zap the errant EOF.

With either of these programs, you first search the file for the Ctrl-Z. A Ctrl-Z is the ASCII character 26 decimal, 1A hexadecimal, or a small left-arrow graphics character. After locating the character, search the surrounding area in the file for useful text. If your text is trapped, the Ctrl-Z must be altered. The safest character to use is the space (32 decimal, 20 hexadecimal). Using the substitute/edit feature of the program, alter the Ctrl-Z(s) and write the file or sector back to the disk.

If you are uncomfortable using DEBUG or NU for this task, ask a knowledgeable coworker or friend for help. When altering disk sectors, you can easily become confused and wreak further havoc by overwriting more information.

DEBUG or NU also can be used to alter non-ASCII files. This task is not easy. Active acquaintance with the ASCII table and the layout of information within the data files and exacting patience are required. Once you know the nature of the corruption, repairing files with a disk-sector editor becomes easier but is rarely easy. Disk-sector editors (and DEBUG) are also poor choices when information must be *inserted*. These programs simply do not facilitate the adding of information in the middle of a file.

The least painful method of salvaging a file may be to restore the backup copy. Before restoring the backup, however, make sure that you keep a copy of the corrupt file. A postmortem examination by a friend or coworker may reveal what happened to the file, or you may discover a program tool that will "heal" the file. Until all hope is lost, keep that copy.

Unerasing a File

Other than dead sectors, accidentally erased files are the most depressing aspect of using a computer. DOS provides no solution to the problem. However, two programs from the Norton Utilities can revive lost life.

One important step must be taken the instant the erasure is discovered: do not record any more information on the disk! This step is essential if the restoration is to be successful.

The reason you must not put any new data on the disk lies in the way DOS erases files. When you instruct DOS to erase a file, DOS does not physically wipe the file from the disk. Instead, DOS takes

two steps. First, DOS changes the first character of the file's name in the directory to an E5 hexadecimal character, a question mark with the most significant bit turned on. The E5 indicates that the file's entry in the directory is no longer valid. Second, DOS marks the file allocation table so that it indicates that these sectors are not in use. This step frees for use by other files the disk sectors that held the first file.

When you erase the file, the file's entry in the directory and its sectors in the file allocation table are freed. The entry in the directory is intact, however, except for the first character in the file name. The chain in the FAT that designates which sectors hold the file is also intact. When you add another file to the disk, the FAT is altered, the directory entry holding the erased file can be reused.

When you add more information to the disk (add a new file or extend an old file), the additional information overwrites the "erased" file, and you lose the opportunity to recover the erased file. The importance of halting any writing to the disk drive holding the erased files cannot be overstressed.

The only DOS program that helps you recover an erased file is DEBUG. Unless you are very familiar with the logical structure of DOS directories, file allocation table, and disk storage, DEBUG is a frustrating exercise in futility. You must manually alter the directory, and you may need to alter the file allocation table. The most experienced DOS programmer avoids this technique whenever possible.

The answer to recovering erased programs is third-party software. Several shareware and public-domain utilities help you recover erased files. The most popular commercial package is from Norton Utilities and consists of two programs for recovering erased files. Because these utilities are friendly and inexpensive, the two Norton programs (QuickUnerase and Norton Utility) are covered in this section.

The directory holding the erased files must exist. If you have already deleted the subdirectory, you must restore it first. See the section on "Unerasing a Directory" in Chapter 11 for further information.

Recovering Erased Files with NU

NU.COM is the catchall program of the Norton Utilities. NU can examine and alter disk sectors, display directory entries, show statistics about disk usage, and "unerase" files. The program is menu driven and simple to use.

The syntax of the command to start NU is

dc:pathc\NU *d: /Dn /Fn /Bn /NOC /P /EBCDIC /EUR /TV*

If you not designate the optional disk drive name, *d:*, NU uses the current disk drive. The switches are described in table 10.11. The switches demonstrate the versatility of NU; however, most people do not need the switches.

Table 10.11
NU Switches

Switch	*Function*
/Dn	Designate the type of *display* to use, *n* is *0* fastest screen updating, used for IBM and close compatibles (the default mode if you do not give the */D* switch) *1* use IBM-compatible BIOS calls for screen updatings (used for close compatibles when */D0* does not work) *2* use ANSI.SYS for screen updating (the slowest mode)
/Fn	*Foreground* color (n is color number from 0 to 15)
/Bn	*Background* color (*n* is color number from 0 to 15)
/NOC	*No color*, for monochrome displays
/P	Display only *printable* characters (for use with the screen-print functions of the computer)
/EBCDIC	For displaying information using the IBM mainframe character set
/EUR	Display foreign (European) characters, the characters with ASCII values greater than 128
/TV	*TopView* mode

After you invoke NU, the program reads the disk's directory, FAT, and several key areas and then displays a menu. You can minimize your work (avoid the need to select the disk drive and subdirectory) by moving into the directory holding the erased files. By being on the proper disk drive, NU does not have to read the disk information from a drive that does not need to be analyzed.

You have three ways to make a menu selection in NU. You may use the up- and down-arrow cursor-movement keys to highlight the selection and then press Enter. You can press the designated special-function key for the selection. You may enter the number of the special function key (for example, 3 for F3). When you are advancing through the program, the highlighted selection on the next menu often is the selection you wish.

The steps to recover an erased file are the following:

1. Select F3 `Recover Erased File`.
2. Select F2 `Select an Erased File to UnErase`. A list of files is displayed.
3. Position the cursor on the name of the file you wish to recover and press Enter.
4. Type the correct character for the first name of the file.
5. Select F3 `Display Information about Erase File`.
6. When the screen displays a message stating that successful unerasing is possible, press any key and choose F1 `Automatically Select All Data Clusters (Best Guess)`. If unerasing is not possible, consult the following discussion.
7. Select F5 `Save UnErased File`.
8. When the message on the screen directs you to press any key, do so. Repeat steps 2 through 7 for each file that must be recovered.
9. Press Escape twice to exit from the program.

NU is "intelligent" when reconstructing files. If no new information has been written to the disk, every erased file is recoverable with little effort. If the displayed information screen states that the file cannot be recovered, the chances are that you are trying to recover an earlier version of the file. Many programs—word processors, for example—create a new version of the file for each revision. Out-of-date entries in the directory can remain for months. In this case, return to step 2 and select a different version of the file.

NU is powerful but maintains a congenial attitude. If you are a newcomer, you will appreciate the menus. If you want a faster but less friendly program, the companion QU program works well.

Recovering Erased Files with QU

QuickUnerase is a less visually oriented file restorer. QU.COM has no menus and few ways to correct mistakes. However, whether

you are recovering a few files or an entire directory, QU is faster than NU.

The syntax of the command for invoking QU is

*dc:pathc***QU** *d:path*

d:path is the optional disk drive and subdirectory name to the erased files. Unless otherwise directed, QU works on the current directory of the current disk drive. Note that a file name is *not* allowed.

QU displays its opening message and indicates how many erased files are recoverable. Then QU displays the names of the erased files one at a time and waits for a response of **Y** to restore the file or **N** to skip the file. If you select **Y**, you are asked for the first letter of the file name. QU then restores that file.

Because some programs perform extensive file shuffling, you should run through the entire list of potentially restorable files first. The first file you encounter may not be the file you wish to unerase. Instead, the first file may be an outdated version that still has a readable entry in the directory. Scanning all possible restorable files helps you discover which file should be recovered. When in doubt, restore the file and examine the contents. If you have recovered the wrong file, erase that file and run NU or QU on another file.

When you are unerasing many files (particularly after a **DEL *.*** command), don't bother to unerase the backup or BAK files. The important files to recover are the later versions. Your time is better spent restoring these files.

QU does not restore incomplete files. You must use NU to reconstruct a partial file. If you use QU immediately after erasing a file, the file should be recoverable.

A Final Thought

When retired Air Force General and internationally famous test pilot Chuck Yeager was on a tour promoting his book, a newsman showed him a picture of two World War II planes in a dog fight. Attempting to gauge Yeager's feelings about war, the wiley newsman asked Yeager for his thoughts on the picture. Yeager's coolly drawled response was "The object is not to be in the picture."

In the same way, your object is not to need to recover erased (or damaged or corrupt) files. Careful, deliberate operations that include checking the typing of commands can eliminate inadvertent file erasing. Inadvertent operations are the major cause of data loss on personal computers.

An erased file, or at least part of the file, usually can be recovered if you act quickly. In only one case is it truly impossible to recover files—physical formatting of a disk. FORMAT wipes floppy diskettes clean. The hard disk physical format program (not FORMAT) does the same for the files on the hard disk. You cannot recover from these operations. (The next chapter includes a discussion of recovering from a *logical* reformatting of the hard disk.)

The best solution for an erased (or damaged or corrupted) file that cannot be restored is periodically backing up critical files. If the backup is timely, restoring and updating the backup file is easier than mending the errant file. If critical files are backed up on a regular basis (daily or weekly), the most information that can be lost is the data entered since the time of the last backup. Using a daily backup means that you forfeit at most one day's work. Using a weekly backup can cost up to one week's work.

Sometimes, however, the only copy of the file is the damaged or corrupted file. In such a case, a disk-sector editor can be used to correct some corrupted files. The DOS RECOVER command may recover the damaged file. Other programs that operate on the entire disk have similar capabilities. If many bad sectors appear, turn to the information in the "Recovering Directories and Disks" section in Chapter 11. You may have more than one file with a problem.

Summary

This chapter has covered file-based operations—ranging from copying files to destroying the contents of files. The following chapter continues the pace established in this chapter but is devoted to operations on directories and disks.

Chapter 11

Directory and Disk Drive Operations

This chapter covers directory and disk drive operations. Directory operations include creating, changing, locating, renaming, removing, and sorting directories. Disk operations include analyzing, repairing, and recovering the logical structure of a disk.

Many disk-related topics overlap. For example, the DOS disk analyzer CHKDSK reports free disk space but also analyzes and repairs some bad directories. CHKDSK also recovers "lost sections" of files. Programs from the Norton and Mace Utilities test the disk and mark bad sectors. These programs and programs that work on a disk-wide basis are discussed in the disk operating section of this chapter.

Because of the blending of different types of operations, you are encouraged to read this entire chapter to avoid missing vital information.

Changing Directories

Changing the current directory is a simple matter. DOS incorporates the CHDIR command into COMMAND.COM. The syntax of CHDIR is

CHDIR *d:path*

or

CD *d:path*

If you don't give a *path* name, CHDIR displays the current directory. If you give a *path*, DOS changes to the last directory named in *path*. *d:* specifies the disk drive.

Although it is simple, the change directory command is not always easy to use. When you are located several subdirectory levels deep on the disk and need to change to a directory in a different part of the hierarchy, typing long directory names can become a tedious chore. Two programs, GO and CED, greatly reduce the typing burden.

Changing Directories with GO

GO.EXE, a program written by Chris DeVoney, first appeared in */c*, a journal published by Que Corporation. An improved version of GO by DeVoney, Tim Leslie, and Alan Stegemoller has been placed in the public domain. The syntax of GO is

*dc:pathc***GO** *name*

or

*dc:pathc***GO** *d:path*

GO uses a second file called GO.DAT. Inside GO.DAT are entries in the form:

name d:\path

name is the symbolic name you give to the directory specified by *d:\path*. When you give GO the symbolic name, GO changes to the disk drive and directory associated with the name. Suppose, for example, that the GO.DAT file held these lines:

```
bin c:\bin
words c:\words
stuff c:\c\stuff
```

Typing the command:

GO STUFF

would make C: the current disk drive and \C\STUFF the current directory. GO.DAT is an ASCII file. Any text editor or word-

processing program in the programmers, or nondocument, mode can be used to add or edit entries in GO.DAT.

GO works like the DOS command CHDIR. You specify a disk drive and path name, and GO changes to the specified directory. Unlike CHDIR, GO also changes to the specified disk drive.

You can see the list of names and directories by typing **GO** without giving a name or path. GO displays the information from the GO.DAT file 23 lines at a time.

With a program like GO on your disk, moving among directories is almost effortless.

GO.EXE and GO.DAT should be placed in the \BIN subdirectory. Directions are included in the documentation for placing GO.EXE and GO.DAT in other directories.

Changing Directories with CED

GO's purpose in life is to be a quick directory changer. CED is a public-domain *command line editor* written by Chris Dunford. CED uses the various cursor keys to edit the current command line or earlier recall and edit commands given to DOS. The editing capabilities of CED are superior to those of DOS.

As a side benefit of CED, you can create synonyms, which are essentially command line macros. After you start CED (the command is *dc:pathc***CED**), you establish synonyms in this form:

CED SYN *name meaning*

name is the name of the synonym; *meaning* is the information that CED will give DOS when you type *name*. The synonym can be a single letter or word and the meaning of the synonym can be one or more DOS commands. When you type the synonym, CED gives DOS the associated command. For example, you can say:

CED SYN DW DIR /W

When you type **DW**, CED makes DOS think you typed **DIR /W** for a wide directory. You can "stack" several commands in one synonym by using the chain character ^ (the circumflex). To establish a synonym that changes to the directory C:\123 and executes the LOTUS menu program, you would type:

CED SYN LOTUS "C:^CD \123^LOTUS"

The double quotation marks are used when you create a synonym that uses special characters, such as the circumflex. The preceding

command creates a synonym named LOTUS, which changed the drive to C, made \123 the current directory, and then invoked the LOTUS program. To use the synonym, you would simply type **LOTUS** at the DOS prompt.

The synonyms can be stored and retrieved in ASCII text files. CED can also write the synonyms to a file for later recall.

A resident program, CED makes its synonym substitutions instantaneously. CED occupies about 25K of RAM, which is space well spent.

CED is a public-domain editor. A commercial version called PCED, which has greater capability, is also available. CED or PCED should be placed in the \BIN or \UTILS directory and should be part of your daily operation.

Creating Directories

When a new directory must be created, the internal DOS command MKDIR (make directory) is used. The syntax is

MKDIR *d:path*

or

MD *d:path*

Either form may be used. *d:* is the name of the disk drive holding the directory; *path* is the name of the directory to be created. If *path* contains a chain of directories (more than one directory name), the last directory named in the chain is created. Also when using a chain of directories, each directory named in the path (except the last directory, the directory to be created) must exist. MKDIR creates one directory at a time.

When a directory is created, its name appears in the parent directory. The single dot (.) symbol for current directory and double dot (..) symbol for parent directory are created automatically by DOS in the new subdirectory.

Erasing Directories

There comes a time when a directory is no longer useful and must be deleted. DOS provides the RMDIR command to remove a directory. The syntax is

RMDIR *d:path*

or

RD *d:path*

d: is the name of the disk drive holding the directory; *path* is the name of the directory to be removed. If *path* contains a chain of directories, the last directory in the chain is removed.

RMDIR is a restrictive command; it will not remove a directory that contains any files or other subdirectories. As a safety precaution, only the entries . and .. are allowed in a subdirectory to be removed.

If you find this precaution annoying, several programs can remove directories that still hold files or other directories.

Erasing Directories with SDARD

An extended directory deleter is SDA Associates' SDARD.COM. Unlike the DOS RMDIR, SDARD will erase or remove any directory (except the root directory which can never be deleted).

SDARD removes directories regardless of files or subdirectories within the intended directory. SDARD will remove complete sections of the directory tree in one command. For these reasons, SDARD is a potentially dangerous command and should always be used with caution.

SDARD offers convenience by allowing you to remove a subdirectory without first deleting all files and subdirectories within the directory. The same convenience can also destroy many unintended files if you are not cautious.

The syntax of SDARD is identical to RMDIR (except that a disk drive/path name to the command may be given):

*dc:pathc***SDARD** *d:path*

d: is the disk drive holding the directory to delete; *path* is the chain of directories to the directory to be removed.

To delete the subdirectory C:\123\FINPLAN regardless of any files or subdirectory within the directory, the command is

SDARD C:\123\FINPLAN

To delete the current directory on the current disk drive, the command would be

SDARD

SDARD always shows the directory to be removed and asks `Are You Sure (Y/N)?`. If you answer *Y* for yes, SDARD removes the files and subdirectories from the intended directory, then deletes the directory itself. As a convenient shortcut, SDARD places you in the root directory of the disk after it has done its work. If you answer *N* for no, SDARD stops, returns to the DOS prompt, and does not change directories.

SDARD can be placed in the \BIN or \UTILS directory with the other members of the FilePaq utilities. If you prefer a shorthand name, rename SDARD.COM to SRD.COM. If you are queasy about having this program readily available, place SDARD in a separate directory not on the PATH. This step, however, is seldom necessary because SDARD incorporates some confirmation before destroying the named directories.

Erasing Directories with ZAP

ZAP.COM is a single-purpose public-domain deletion program. The syntax for ZAP is

*dc:pathc***ZAP directory_name**

ZAP's syntax is limited. ZAP only works on subdirectories in the current directory on the current disk drive.

ZAP is quick and dangerous. It does not give any error messages or warnings. ZAP zaps the subdirectory and all files and further subdirectories in the named subdirectory. For this reason, ZAP must be used with caution. You can ZAP a subdirectory that may take hours to reconstruct.

A problem you may encounter occasionally is that you cannot remove your current directory. You must move out of the directory to delete it. (As already mentioned, SDARD does not have this limitation.)

Confident users can place ZAP in their \BIN or \UTILS directories. If you have any reservations about placing ZAP in a directory that is on the DOS PATH, place ZAP in a separate directory and specifically invoke the command. My personal preference is to use ZAP over SDARD.

ZAP is faster, but SDARD is more versatile and requires confirmation, which can help you avoid a painful reconstruction if the wrong directory name was given.

For information on recovering an erased directory, see the section "Recovering Erased Directories" later in this chapter. For information

on recovering erased files, see the section "Unerasing Files" in Chapter 10.

Locating Directories

After long periods of using hierarchical directories, any hard disk owner can forget the structure of the directories. Just as files can be lost, directories can be lost. To help find them, several programs let you see the hierarchical structure.

The DOS TREE command is one such program. The full syntax for TREE is covered in the section "Locating Files" in Chapter 10. TREE prints each directory and each directory's subdirectory. The output of TREE can be redirected to the printer, providing a printed road map that is helpful to others who use your computer.

Other programs mentioned in Chapter 10 also can be used. **SDADIR /DO /T**, for example, will traverse the directory structure and print only directories.

The best program to date for locating directories is a small (1K) public-domain program called VTREE.COM. VTREE displays a visual map of your directory structure and generates a more appealing output than TREE's.

The disadvantage of VTREE is that VTREE uses some of the PC's special characters (lines and boxes). Unless your printer handles IBM graphics character sets, the graphics characters will be mangled when you redirect VTREE's output to the printer.

VTREE's syntax is simple:

*dc:pathc***VTREE** *d:*

The single argument to VTREE is the disk drive to display (*d:*). VTREE.COM is a delightful program that quickly displays the directory hierarchy. The program belongs in the \BIN or \UTILS directory.

Recovering Erased Directories

Generally, you cannot accidentally erase a directory using DOS commands. The RMDIR command does not remove a directory that contains files. But outside super delete programs can wipe out directories and files in one operation. When this power is unintentionally used, you *must* recover the subdirectory before you

recover the erased files. The subdirectory holds the entries from the files. If you haven't successfully recovered the directory, the process of recovering the files is an electronic form of finding a needle in a haystack.

The only directory unremover discussed in this section is Unremove Directory (UD.COM) from the Norton Utilities. The syntax for UD is

*dc:pathc***UD** *d:path*

The single argument to UD is the disk drive and path to the *parent directory* of the erased subdirectory. Do not give the path including the erased subdirectory. The subdirectory no longer exists to DOS. You must operate on the parent subdirectory to recover the removed child.

UD works in two phases. In the first phase, UD shows a list of potentially recoverable directories. To select the directory to unremove, you answer *Y* (yes) or *N* (no) to the prompt. As with erased files, the first character of the erased subdirectory name has been altered and must be restored. After you give the first character, UD begins to rebuild the subdirectory.

Because of the way DOS stores directories, the program enters the second phase to mark which files are valid. You may need to specify what part of the data file (subdirectory) is valid.

Groups of file names are presented and you answer *Y* or *N* when prompted whether to include or exclude the groups. Names that look like nonsense are probably stray disk sectors and should not be included. Once you have selected the groups, UD restores the subdirectory.

To finish restoring the files within the directory, however, you must run a file uneraser, such as Norton's QU or NU. See Chapter 10 for more information on QU and NU.

Like all other Norton Utilities, UD should be located in BIN, UTILS, or in a separate directory on the DOS PATH.

Renaming Directories

After your hierarchical directory is established, there will come the occasion when you wish a directory had a different name. Internally DOS has the facilities to rename a directory; externally the command interpreter (COMMAND.COM) does not allow you to rename a directory.

Without an outside tool, you must create a new directory with the desired name, move all files from the old directory to the new directory, and delete the old directory. If the directory to be renamed has subdirectories, the task is further complicated. The subdirectories also must be copied.

RENDIR.COM is a public-domain program that renames directories. The syntax is

*dc:pathc***RENDIR old_name new_name**

The first name is the current name of the directory. The second name is the new name for the directory. Disk drive and path names are not allowed. You must be in the parent directory of the directory to be renamed. Although the command is limited, RENDIR is useful for those occasions when you want to rename a directory the easy way.

There is one caution. If you rename a directory, you may need to revise several batch files or programs with the new name. The old directory no longer "exists." Programs that look for the old directory name will report errors. Batch files that depend on the old directory name will fail.

RENDIR should be placed in the \BIN or \UTILS subdirectory.

Sorting Directories

DOS's DIR command presents file names as they occur in the directory. Extended directory programs, such as SDADIR, can present a sorted list of files. Another sorting alternative is the Directory Sort program of the Norton Utilities. DS.COM can sort a directory by a multitude of criteria.

The syntax of DS is

*dc:pathc***DS** *NEDTS d:path /S*

d:path is the disk drive and path name of the directory to be sorted. *NEDTS* is the criteria for the sort:

N	sort by root *name*
E	sort by *extension*
D	sort by file *date*
T	sort by file *time*
S	sort by file *size*

One or more criteria for the sort must be given as a single word. Each criteria can be preceded by a minus sign (-) to reverse the sort—to sort in descending rather than ascending order, for example.

The single switch */S* designates that all *subdirectories* should be sorted (traverse the subdirectory tree).

To sort the current directory by file root name and then by extension, the command is

DS NE

To sort by descending extension and then by ascending root names the command would be

DS E-N

To place the most recent files at the top of the list in C:\123\REPORTS the command is

DS -D-T C:\123\REPORTS

The most frequent use of DS is to sort by name and then by extension (NE) or by extension and then by name (EN). Sorting by name and extension places files with the same root name together, a procedure that is often helpful in identifying a set of files or a missing file. Sorting by extension and name helps keep files with a common purpose (common extension) together.

DS quickly sorts the directory in any order you desire. You need not run an extended directory program to see the files in sorted order. You can quickly identify files in a set or determine which files are missing.

The only potential astonishment with DS is that subdirectory names are also sorted. Because most subdirectory names do not have an extension, these subdirectories bubble to the top of the list when ascending sorts are chosen.

DS directly manipulates directories. The only caveat associated with DS is that you should not interrupt the program while it works. The operation takes less than three seconds, but pressing Ctrl-Break or shutting the system down during that time can damage the directory.

DS is a useful utility that can quickly rearrange the order of the directory. DS should be placed in \BIN, \UTILS, or in a separate PATH directory that holds other Norton Utilities files.

Analyzing and Repairing Damaged or Reformatted Disks

This section looks at two distinct problems: damaged disks and unintentionally reformatted disks. Either problem can cause catastrophic data loss. The extent of damage depends on the area or areas affected. A single bad sector in an unused portion of the disk can be detected and "eliminated" without loss. But one or more bad sectors in the critical disk housekeeping areas can be disastrous, as can inadvertent reformatting of the hard disk.

Disk damage is not always physical. For example, key housekeeping areas (a directory or file allocation table) could be written properly but with incorrect information. Interrupted writing of correct information to the housekeeping area can be another problem. The result of either problem is corrupted housekeeping information that can be as devastating as a bad sector. Corrupted housekeeping areas are the result of *logical* damage. Bad sectors are a sign of *physical* damage.

You can split hairs about which malfunctions are classified as physical damage and which malfunctions are classified as logical damage. Obviously, physical damage to a subdirectory or FAT can cause logical damage.

This is the working definition: if the damage relates to something created by the physical formatting program, the damage is physical. If the damage relates to something established by FDISK or FORMAT, the damage is logical.

Regardless of definition, this chapter focuses on the diagnosis and repair of the problem. Discussion of these topics is more meaningful when problems arise than discussing whether a particular malfunction is physical or logical.

Logical damage such as corrupted directories and file allocation tables can be completely or partially mended using DOS's CHKDSK or RECOVER programs. You may recall that the RECOVER command also can be used to retrieve a single file. A second use for RECOVER is reviewed in this chapter. Finally, the discussion covers the Mace Utilities functions that can recover unintentionally reformatted hard disks.

To repair physical damage within a nonhousekeeping file (normal files), you can use DOS's RECOVER or the REMEDY program from the Mace Utilities. The use of RECOVER with a single file is covered

in Chapter 10. REMEDY is covered in this chapter. Damage to other areas can be diagnosed and managed with other tools presented here. These tools include the Norton Utilities' Disk Test and the HTEST/HFORMAT package by Kolod.

Analyzing and Repairing the Logical Structure with CHKDSK

The first program covered in this section is the DOS check disk (CHKDSK.COM) program. CHKDSK's functions include analyzing use of the disk, displaying total memory and free memory, repairing minor damage to directories, and restoring lost disk clusters. CHKDSK is most frequently used to determine how much space on a disk (floppy or hard disk) is available and how much memory is free. As you'll see, CHKDSK performs much more.

CHKDSK limits its activities to the "logical" structure of the disk. CHKDSK does not examine the underlying physical storage. This means that CHKDSK does not attempt to "fix" or mark bad sectors. Other programs must be used for that purpose.

The complete syntax of CHKDSK is

*dc:pathc***CHKDSK** *d:path**filename.ext /F*

Including or excluding the optional information causes CHKDSK to take different actions. *d:* is the name of the disk drive to analyze. If no drive name is given, the current disk drive is used.

If you give a path and file name, CHKDSK checks the named files for continuity. Wild cards are permitted in the file name. CHKDSK reports whether the named files are contiguously stored; that is, stored in consecutive sectors. For this reason, view the *path**filename.ext* part of the command as a group. If you do not give this information, CHKDSK analyzes the disk drive.

CHKDSK always reports the following information:

- The volume label with its date of creation (usually the date the disk was formatted)
- Total disk space
- The number of hidden files and amount of storage (actually hidden and/or system files)
- The number of directories and amount of storage
- The number of normal (called user) files and amount of storage
- The amount of space taken in bad sectors; the amount of disk free space

- The amount of DOS RAM memory (not including expanded or extended memory, which is not stated) and the amount of free memory

If you give a path-file name, CHKDSK either reports that all files are stored contiguously or names the noncontiguous file(s) and the number of groups the file is stored in.

If CHKDSK detects errors, the program goes through the motions of repairing the logical structure of the disk and reports the results. But CHKDSK does not actually repair the disk unless the */F* (*Fix*) switch is set. When */F* is used, CHKDSK will repair the disk's structure.

Because CHKDSK's repairs may make parts of your files difficult to recover, you should usually run CHKDSK without setting the */F* switch. Note any problem that CHKDSK reports and then rerun CHKDSK with the */F* switch set. Following this procedure gives you an opportunity to rescue files that have major problems (by copying them to another disk) *before* CHKDSK makes its attempt.

CHKDSK's actions ensure the correct interrelationships among the directory, the FAT, and the disk itself. Remember that a file's directory entry contains the file's size and a pointer to the file's starting cluster in the file allocation table. Unlike other operating systems, the FAT is the key to locating files. If the FAT is damaged or inconsistent, your files are at risk.

CHKDSK analyzes lost clusters. Lost clusters are orphans of the disk, marked in the FAT but not in use by any file. Interrupted disk activity is usually the cause of these wayward clusters.

Here is one way disk activity may be interrupted. Programs issue several distinct signals to DOS when they operate on files. They must explicitly command DOS to open or create a file. As data is written to the file, DOS records the data and updates the FAT. When the program is finished with the file, DOS is told to close the file. The closing process ensures that all housekeeping information is written to the disk.

Before a file is "officially" closed, certain information may be housed in a DOS buffer. If anything happens to the computer before the file is closed, DOS may not complete the required housekeeping and, *voila*, clusters are orphaned. The "anything" can be a power failure, a program lockup, an errant (faulty) program, or an abort of a program (the program is interrupted before it closes the file).

Another potential cause of orphaned clusters is corrupt information written to a directory or FAT. The offender is usually

malfunctioning hardware or a malfunctioning program that bypasses DOS. This problem occurs very, very infrequently.

Lost clusters may contain useful or useless information. Most programs, such as word processors or spreadsheets, create new versions of a file when the file is revised. Interrupting the programs and DOS when the new file is being written is not usually a problem. The old file is intact. If the interrupted program is altering information or extending the original file (rather than creating a new file), the lost clusters hold useful information.

In most cases, CHKDSK can fix the problem by creating directory entries for each contiguous set of orphaned clusters. When the */F* switch is given, the clusters are assigned the name `FILExxxx.CHK` in the root directory. The `xxxx` is a sequential number from 0000 to 9999. You can examine each file and decide if it can be safely deleted. Most frequently, you will find that the original file is intact and the orphan can be removed.

When CHKDSK finds lost clusters, you get the following message:

```
xxxx lost clusters found in yyy chains.
Convert lost chains to files (Y/N)?
```

The `xxxx` is a number between 0000 and 9999 and the `yyy` is a number between 000 and 999. If you answer *Y* to the conversion prompt, CHKDSK will go through the repair motions and report the statistics. If the */F* switch is set, CHKDSK creates the files from the orphaned clusters and reports the results. If you answer *N* to the prompt and the */F* switch is set, CHKDSK repairs the FAT by marking the orphaned clusters as free. Either way, the directories and FAT are consistent.

Lost clusters are not a major problem. They can be cured easily by CHKDSK. If, however, you create orphans on a frequent basis, analyze your programs and your actions. Something (a program that does not close files) or someone (a person who prematurely aborts programs) is the cause.

Other problems that CHKDSK handles—corrupted directories or FAT entries—are a cause for more concern. The problems range from two files "owning" the same space on the disk (each file's entry in the FAT indicates that the cluster belongs to that file) to file sizes (based on the file's directory entry) that are larger or shorter than the number of clusters allocated in the FAT.

If one of these problems arises, first try to copy the affected files to another disk (or pray that your backups are current) and then run CHKDSK with the */F* switch set for repair. CHKDSK seeks the

shortest route to repairing an inconsistency, an action that often leaves the file's contents at risk. If you try to make a copy of the file first, you may be able to capture the file's contents whole.

Remember, you should use CHKDSK first to check the disk (without the */F* switch). Then run CHKDSK with the */F* switch. If CHKDSK reports anything wrong other than lost clusters, copy affected files to another disk before you run CHKDSK the second time.

Repairing the Logical Structure with RECOVER

As mentioned earlier, DOS's RECOVER has two forms. The first form is used to recover one file. The second form attempts to recover all files on a disk. In this section the second form is discussed.

The second form of RECOVER fiddles with the directory of the disk. This form of RECOVER is *dangerous*. RECOVER should be used only when the root directory of the disk is damaged and files are trapped. Don't run this form of the command if any other alternative is available. Once DOS has "fixed" a directory with RECOVER, you may spend hours trying to recover.

The syntax of this form of RECOVER is

*dc:pathc***RECOVER** *d:*

The *d:* is the optional disk drive name and is given if the command is to be used on a different disk drive than the current drive.

DOS runs down the FAT and creates files in the root directory in the form `FILExxxx.REC`, `xxxx` being a four-digit number from 0000 to 9999. When the root directory fills, RECOVER stops. To recover more files than your root directory holds, you must take the next step (examining the new files, deleting unneeded files, and then rerunning RECOVER).

After RECOVER has performed its work, your detective work begins. Your task is to determine what file holds what information, what files should be salvaged, and what files should be discarded. For ASCII files, your tools are TYPE, MORE, and similar programs. For nonASCII files, you need a file-content displayer such as DEBUG and NU (from the Norton Utilities) and the public-domain DUMP program. You have no information on original file names after running RECOVER; that information is lost.

Recovering from running RECOVER on a 360K diskette with twenty files has taken several hours. The task of recovering a 20M or larger hard disk with thousands of files is, understandably, larger. I strongly recommend that you use any other backup, copy, or other method to recover files before you use RECOVER on a disk drive. RECOVER should be viewed as a last resort.

Analyzing and Repairing Bad Sectors

A bad sector, a reflection of physical damage, can have several potential causes. Damaged physical media, poor formatting, and a flaw in the recording are possibilities. Strangely, two other factors are more critical than the cause of bad sectors: the timing of the problem's discovery and the location of the bad sector. If a bad sector is caught by a formatting or disk checkout program, for example, the sector is marked as bad and no information is placed in the sector. Similarly, if the location of the affected sector is non-critical, one file, at most, may be in jeopardy.

The converses are bad omens. Sectors that "go" bad after the disk is formatted can indicate that more sectors may go bad in the future. If the location of the bad sector is critical, a file, a directory, or even the entire disk can be lost.

To keep the discussion in perspective, however, note that a bad recording of information occurs only about once in every ten million hard disk write operations. When you use an error-checking and self-correcting (ECC) coding of the information, the potential for loss is only one in one billion. A "natural" occurrence of a bad sector is rare. However, electronic failures or abuse can boost the probability of failure.

What Is a Bad Sector

Bad sector is a sweeping term, encompassing both physically and logically damaged sectors. Unlike floppy diskettes, few new hard disks are without physically damaged portions. Because of their narrow acceptable tolerances, most hard disks have a few physically damaged sections. These sections are detected during factory testing and are marked on the outside of the disk drive.

You may notice that the hard disk physical formatting programs ask for this list. The program does not bother to test these bad

sectors. Physical formatting programs, which can take two to four hours, usually find any additional areas the factory has missed.

Note that to testing and formatting programs there is no such thing as a "bad sector." The entire track holding a "bad" sector is marked bad. Hence, hard disk bad tracks are designated by the cylinder and head (which form a track). Many hard disks devote a small section of the disk to the storage of a bad track table. The disk's internal electronics prevent the operating system from "seeing" and using these bad tracks.

Floppy diskettes do not use the same scheme. DOS handles floppy diskette bad-sectors by marking the sector in-use in the FAT. After absolute formatting, DOS uses the same in-use scheme to "hide" bad sectors on the hard disk. The effect of the scheme is that DOS does not attempt to use the faulty areas when writing information to the disk.

At the time formatting is complete, all bad tracks should have been identified. Later, as information is stored, the information is placed on good sectors.

Sectors that go bad after formatting are a problem. Something has caused the bad sector. Development of bad sectors on a hard disk indicates a physical shock or logical problem. Physical damage may be caused by shock to the drive or by worn mechanical parts, such as a spindle. The term "head crash" has shuddering significance.

The more frequent cause of emerging bad sectors is disk electronic problems: the housekeeping information or the data was not recorded properly or not recorded at all. A bad sector emerges.

Amazingly, DOS's FORMAT does not always detect all hard disk bad sectors. FORMAT does test each sector, but does not vigorously exercise (test many times) each sector. Bad sectors can slip by DOS. This is one of the reasons why disk drives failed on some IBM Personal Computer XTs under DOS V2.1 and on Personal Computer ATs with the CMI disk drives.

The Location Is the Story

The impact of a new bad sector's location ranges from benign to malignant to critical. A bad sector that is not used because it has been detected by FORMAT or any disk test program is benign. With no data recorded at the spot of the bad sector, no information is lost.

An emerging bad sector can be malignant. If DOS later attempts to place information in the sector, data could be lost. If the bad

sector develops where part of a file is stored, part of the file will be unretrievable. If a bad sector develops where a subdirectory is stored, multiple files can be lost. Each scenario has different shades of malignancy.

The worst location for a bad sector is on cylinder 0. All disk drive manufacturers certify that cylinder 0 is error free on each disk drive. This guarantee is critical, because the most vital housekeeping information is stored on cylinder 0. To explain the possible devastation of a bad sector in this area, a discussion of cylinder 0's use is necessary.

Starting on side 0 and head 0 at sector 1 (the first sector of the disk) is the master boot record. Sharing this sector with the master boot record is the disk partition table.

A hard disk can be subdivided into several "logical" disks, one or more for each operating system. The disk partition table holds four entries, one for each of the four possible partitions. The information for each entry, in order, is

- Whether this is the start-up (active or bootable) partition
- The starting location (cylinder, head, and sector) of the partition
- The operating system that owns (uses) the partition
- The ending location (cylinder, head, and sector) of the partition
- The number of disk sectors before the start of the partition
- The number of disk sectors owned by the partition

When you boot from the hard disk, the master disk boot record is first loaded. Its task is to read the disk partition record looking for the active partition.

In turn, the master disk record loads the first sector of the active partition, which must be the boot record for the owning operating system. In essence, two bootstrap programs are loaded when the hard disk starts, one to locate the correct boot record, the second to start the correct operating system.

If the entire disk is devoted to DOS, the DOS boot record is on the second sector of the disk, and the first DOS file allocation table starts on the third sector. After the file allocation table comes the root directory, and then the file holding the BIOS of the operating system, MSDOS.SYS or IBMBIO.COM. The DOS bootstrap program has little intelligence; the BIOS file must be stored on contiguous sectors or the boot fails.

On 20M and larger disk drives, both cylinder 0 and head 0 hold the master boot record, the partition table, and part of the first FAT. Head 1 contains the remaining portions of the first FAT and the second FAT, which extends through head 3. Head 3 starts the root directory. Most 20M disk drives have four heads (two platters). The root directory continues onto cylinder 1, heads 0 and 1. Heads 2 and 3 of cylinder 1 hold the BIOS. On most 30M and higher disk drives, cylinder 0 may also hold the entire root directory and BIOS, or the files may continue to cylinder 0.

Imagine the effects of a bad sector developing on cylinder 0. The best effect you can hope for is losing an unused part of the root directory. DOS will display an error message that the directory cannot be fully read, but that you can make backup files and reformat the disk. The worst effect is the loss of the master boot record or partition table. In that case, the entire disk may be lost.

If the partition table or master boot record is destroyed, you must run FDISK to restore the table. Unfortunately, this disk recovery process can be disrupted by a small change in FDISK between DOS V2 and V3. FDISK V2 warns that changing the partition table will destroy data. This was not always true. FDISK rewrote the master boot record and the partition table, but did not touch any other part of the hard disk. The root directory and FAT stayed intact. When FDISK V3 warns data will be destroyed, it means it. FDISK V3 rewrites the remainder of cylinder 0, also *wiping* the FAT and root directory. The data on the disk is lost.

Did you know that the file allocation table is the most frequently used portion of the disk? Given an errant write that occurs randomly, what cylinder is most likely to be affected? The answer is the critical cylinder 0. An errant write to cylinder 0 is usually fatal. When cylinder 0 is "trashed," the disk suffers the same fate. The files on the disk usually are unretrievable.

Periodic testing of the disk helps detect new bad sectors and prevents DOS from writing to these sectors. Several programs are available for this testing. Both Disk Test from the Norton Utilities and EXAMINE from the Mace Utilities hunt and mark bad sectors. HTEST from Kolod Research does not mark bad sectors but performs a more extensive analysis of the disk than Disk Test or EXAMINE.

These programs, which take from thirty minutes to four hours to run, can be used every month or so to detect bad sectors. At the first sign of failure, however, back up the disk and run one of these programs *immediately*.

In following sections this chapter discusses the Mace Utilities' UNFORMAT function, which can heal disks with damaged cylinder 0s.

Using Disk Test To Diagnose Bad Sectors

The Norton Utilities Disk Test program helps detect bad sectors and can mark bad sectors from the hard disk. DT.COM can diagnose two different areas of the disk, those portions used by subdirectories and files and those portions not yet in use. The syntax of DT is

*dc:pathc***DT** *d: /F /D /B /LOG*

d: is the name of the disk drive to test. The switches are

/F	Test only those areas in use by *files* and subdirectories
/D	Test the entire *disk*, regardless of use
/B	Perform *both* tests
/LOG	Format the output for redirection to a printer or disk file to produce a *log* of activity

DT reports any errors found and renders an opinion. For bad sectors that are already hidden in the FAT, DT reports "no danger." If the bad sector is not marked in the FAT but is not used by a file either, DT reports "danger to come" and marks out the sector in the FAT.

If the bad sector is part of a file, DT reports "danger now" and automatically enters the file-test portion after completing the full disk test. DT will give the location of the bad sector, but not take any other action. You must copy the file manually or run DOS's RECOVER.

The Disk Test file should be placed in the \BIN or \UTILS directory, or in a separate directory on the PATH.

Analyzing and Repairing Bad Sectors with the Mace Utilities

The DIAGNOSE and REMEDY options of the Mace Utilities can also analyze and repair bad sectors. DIAGNOSE simply looks for and reports on bad sectors; REMEDY diagnoses and "fixes" bad sectors.

You follow the same general procedure to use both options. After starting the Mace utilities, you select special-function key F3 for DIAGNOSE, or F5 for REMEDY. You are asked which disk drive to

analyze and which drive is to receive a disk-based copy of the diagnostic report. If you suspect your hard disk is having problems, send this report to an empty, formatted floppy diskette. The report may take up 50K to 200K, depending on how many times the diagnostic should run and how many bad files you have.

Both DIAGNOSE and REMEDY read each disk sector. Upon finding a bad sector, DIAGNOSE reports the damage and states what file, if any, is affected. REMEDY reports the damage and moves the remaining parts of the file to a good location of the disk. Then it marks the bad sector in-use in the file allocation table so no other file uses the bad sector.

The Mace Utilities must be installed in a separate subdirectory, preferably \MACE. The Utilities generally may not be executed from another directory, and \MACE should not be on the PATH. You can, however, use this batch file to invoke the Utilities:

```
CD C:\MACE
MACE
```

Analyzing and Formatting Bad Sectors with HTEST/HFORMAT

Kolod Research developed HTEST/HFORMAT (now marketed by Paul Mace), a useful group of utilities designed to find many hard disk operating problems. HTEST is a comprehensive disk diagnostic program. HFORMAT is a thorough physical (low-level) formatting program. Of the programs mentioned in this section, HTEST has the most thorough disk diagnosis capabilities, but does not fix bad sectors. HFORMAT does not fix bad sectors per se; it physically reformats the hard disk. The two provide a useful brute force method of reformatting selected hard disk cylinders.

HTEST and HFORMAT have outstanding flexibility. The operations can be simple (test or format the entire disk) or complex (test or format certain disk sections using a specified movement or data pattern). The user has flexibility in what operations should be performed and the number of times operations should be repeated.

The general pattern would be to use HTEST to test the hard disk and HFORMAT to physically reformat the disk if necessary. The testing can be nondestructive (HTEST tests by reading information) or destructive (HTEST writes and reads information).

The test can be confined to certain sectors, cylinders, or recording heads. The recording head movement pattern can be run

from the beginning of the disk to its end, from the center of the disk to the outside cylinders, or in an almost random pattern. The various movement patterns exercise the recording arms to ensure that the recording arm assembly and the drive electronics are functioning properly.

HFORMAT is the physical hard disk formatter. Physical formatting normally is performed before you receive the hard disk, but must be performed again if the disk "loses" its format or if you wish to change the interleave factor. Losing format means the track and/or sector identification (each track and sector is marked with a number) has been lost. This occurs when an electronic malfunction rewrites the information. The only recourse is to reformat the damaged tracks. DOS's FORMAT cannot perform the same work.

HFORMAT can format the entire disk or only the offending tracks and can test the disk as extensively as HTEST. If, however, HTEST finds an offending sector that HFORMAT missed, you can use HFORMAT manually to mark the sector. The sector enters the bad-track table and is hidden from DOS.

Generally, using HFORMAT to fix bad sectors is a last-resort method. HFORMAT should be used only when an error occurs on cylinder 0. This topic is discussed in the next section. HFORMAT can be used when the hard disk is first received or any other time physical formatting is required.

Usually, you will want to use the nondestructive testing of HTEST. This testing should uncover most problems. However, backing up the information on the disk before running HTEST is considered both good practice and preventive medicine (in case you accidentally choose destructive testing).

If you use the destructive test (the write test) of HTEST or use the HFORMAT program, you must back up your disk's information before you run the program because this HTEST and HFORMAT work is guaranteed to destroy your information. If you reformat the entire disk with HFORMAT, you also must rerun DOS's FDISK and FORMAT before you can use the hard disk again.

Because these programs extensively test the disk, each program takes between thirty minutes and four hours or more to run. The time depends on the type of disk drive tested (number of cylinders, access time of the drive, and number of times the tests should repeat). Although testing is tedious, the time is well spent. Comprehensive testing is more likely to reveal bad sectors than less extensive tests.

The documentation for HTEST and HFORMAT is as thorough as the programs. Newcomers may find the documentation slightly intimidating, but others, from advanced novices through experienced users, will find it excellent. Many pages are devoted to describing each program's possible results and to explaining in tutorial fashion the ins and outs of disk storage and performance. The obvious intent is to help you understand the theory as well as the operation of the programs. The documentation is excellent reading.

There is just one quirk to these programs you should note if you use two hard disk drives. The programs unconditionally label the first disk drive C. One of my associates uses two hard disk drives, but devotes the first hard disk to the XENIX operating system; the second disk drive is used by DOS. To DOS, the second disk drive is drive C and the first disk drive is "invisible."

HTEST and HFORMAT, however, see the XENIX drive as C and the DOS disk drive as D. Because of this nonstandard setup, HTEST/HFORMAT could operate on a different disk drive than intended—with unpleasant results.

If I hadn't experienced as many problems as I did with an AT, I probably wouldn't even know about Kolod Research's programs. IBM's Advanced Diagnostics provide some of the features of the Kolod programs, but the Advanced Diagnostics are not as comprehensive. And they are not provided to all customers (the Advanced Diagnostics must be purchased separately). Kolod has neatly filled the gap left by IBM.

I consider HTEST/HFORMAT a "life-preserver" program, a needless expense until hard disk problems appear; then the package is absolutely essential. Unfortunately, judging when hard disk problems will occur is impossible. The various-movement testing of HTEST can help identify impending problems, however. For this reason, running HTEST every one-to-three months has merit.

The safety play is to purchase the package before hard disk problems occur. The riskier play is to wait for the first signs of a problem, then rush for the package. And if you find diagnosing hard disk problems distasteful or too challenging, let your dealer do the work when your hard disk shows any sign of failure.

Additional programs in the HTEST/HFORMAT package include HOPTIMUM, a program to determine experimentally the optimal interleave for the disk drive; and GETSEC and PUTSEC, absolute disk sector read and write programs that can restore a damaged disk. The package also includes a program that parks the disk recording heads.

Because of the destructive nature of HTEST and HFORMAT, I recommend that both programs be kept in a directory not on the PATH or not on the hard disk at all. Because the programs are infrequently used, the HTEST/HFORMAT set is best kept on a floppy diskette in an accessible area.

Recovering a Damaged or Formatted Disk

Earlier, you saw that an erased file or directory can be recovered *if* you have the right tools. If nothing has been added to the disk after the erasure, the file is recoverable. If information has been added, the file still *might* be recoverable because only the directory and FAT are altered.

On the other hand, you "know" that, once you reformat a disk (fixed or floppy), that is it! Data is wiped out irrevocably.

Of course, accidentally reformatting a hard disk isn't easy. All DOS V3 versions that recognize hard disks scream:

```
WARNING! ALL DATA ON NON-REMOVABLE
DISK DRIVE C: WILL BE LOST
Proceed with Format (Y/N)?
```

and insist that you reconfirm (or reexamine) your intentions. IBM's FORMAT V3.2 requires the user to specify a disk drive name and give further confirmation by typing the disk's volume label. Accidental reformatting is now even more difficult than it used to be.

Accidental reformatting is most likely to occur where many different users operating at different levels of expertise use the same machine. You can "hide" FORMAT as was shown in Chapter 7, but no matter what safeguards are employed, someday, somehow, a disk that shouldn't be will be reformatted, accidentally or maliciously.

When this happens, isn't your only alternative to pray your backups are up-to-date? Aren't you dead in the water?

Not necessarily. The Mace Utilities can save the bacon with a program that relies on the fact that FORMAT does not truly format a hard disk.

An earlier discussion mentioned that formatting is a two-step process, physical and logical. Physical formatting involves establishing the tracks and sectors of the disk and writing the necessary disk housekeeping information. Logical formatting involves writing the boot record, the FAT, the root directory, and DOS files (if FORMAT */S* was used).

FORMAT physically and logically formats a diskette, but it only logically formats a hard disk. The disk trashing that FORMAT does during the "formatting" of a hard disk is a brief test for bad sectors. Physical formatting is left to a different program.

If you have operated under the assumption that FORMAT wipes information from the hard disk, you are partially correct. The DOS boot record, FAT, and root directory are reestablished (meaning that the previous information is destroyed). But all other information is intact! The additional time that FORMAT takes is to perform a rudimentary test of the hard disk's individual sectors.

The Mace Utilities can restore your vital files. Mace's two unformat procedures handle reformatted disks, and its RESTORE BOOT SECTOR can handle hard disks with "blown" cylinder zeros.

I have two qualms about the programs. If you are not prepared, the Mace procedures cannot restore all information on the disk. The other qualm is that the programs may give you a false sense of security. You may get complacent about backing up or get sloppy in your commands to DOS. These programs are not a substitute for a careful and well-managed program of regular backups.

You also should heed this caution. COMPAQ's MS-DOS V3.1 and one or two versions of AT&T's MS-DOS V2.11 (AT&T has released several versions of V2.11) actually *do* perform a destructive format of the entire hard disk. The FORMAT program writes "zeros" into every sector of the disk. In other words, running FORMAT on these two versions of DOS is *final*. The Mace Utilities cannot recover information from a disk formatted by COMPAQ V3.1 or AT&T V2.11 DOS.

Unformatting a Disk with the Mace Utilities

The Mace Utilities include two different modes of the unformatting operation. A succinct quotation from the manual summarizes when to use the first mode: " . . . If you are already in trouble, i.e., someone has already reestablished-formatted the hard disk and you are sweating." The second mode, which is discussed shortly, can be used to restore a damaged cylinder 0.

In its first mode, the UNFORMAT option reconstructs all subdirectories and their files. The first mode cannot construct the files in the root directory, and noncontiguous (fragmented) files may contain spurious information (i.e., junk). However, level one subdirectories are restored to the root directory. Faced with the complete loss of information versus recovering most of your

information, using the first mode of UNFORMAT is more attractive than not running UNFORMAT at all.

The UNFORMAT option is located in the utilities menu (special-function key F2, followed by F10). After you unformat the disk, you must copy as many files as you can onto another disk, reformat the unformatted disk, and then copy the files back. UNFORMAT cannot properly restore all files. Mace leaves some files in a "state of disgrace," meaning that two files may contend for the same disk cluster. To eliminate this problem, the disk must be reformatted.

The second unformatting mode is more ideal. This mode requires you to install the Mace Utilities as a preventive measure, *before* you get into trouble. After you have installed and invoked the programs, you tap special-function key F8 to create a disk-based file called BACKUP.M_U. BACKUP.M_U contains a copy of the master boot sector of the hard disk, the DOS boot sector, the file allocation tables, and the root directory.

After the file is created, you must run the RXBAK.EXE program periodically. RXBAK can be run independently, unlike the remaining Mace Utilities that are enabled through the Mace menu. The safest and most secure procedure is to add a line to your AUTOEXEC.BAT file. Using this technique automatically updates the BACKUP.M_U file whenever you start DOS. It also gives you the best fighting chance to restore your disk. Assuming that the Mace Utilities are installed in a directory called \MACE, the line to add to your AUTOEXEC.BAT file would be

\MACE\RXBAK

The BACKUP.M_U data file that holds the disk information is stored on the hard disk. Although it seems foolish to store this file on the same hard disk that might be formatted, the method is safe and sane. The Utilities physically store the file near the end of the disk. Because FORMAT destroys the information at the front of the disk, the possibility of destroying this vital file is very slight. Because the file is stored in a "fixed" location, the Utilities can easily pinpoint the file's location and use the file as a Rosetta Stone to recreate the disk structure.

To restore the disk, use special-function key F2 to select the utilities menu; then press F5 to RESTORE BACKUP. You may need to run CHKDSK to clean up files that were altered or created after the last running of RXBAK.

Repairing a Damaged Cylinder 0 with the Mace Utilities

You get the second worst feeling in the world when you notice your hard disk has been reformatted. The worst feeling is when, after specifying the hard disk, you see the DOS message:

```
Invalid disk specification
```

Or when you attempt to start DOS from the hard disk and see any of these messages:

```
Bad Partition Table
Disk boot failure
Drive not ready error
```

Something is seriously wrong with the hard disk. The first message (bad partition table) means the master boot record has been destroyed. The second message may mean that the disk-based copy of the DOS system files (MIO.SYS and MSDOS.SYS or IBMBIO.COM and IBMDOS.COM) has been corrupted. The third message usually means an electronic failure exists in the disk drive or adapter board.

Handling the messages in reverse order, the third message may mean a trip to the repair shop is in order. Before you check your disk into a repair station, you should try running a diagnostic program (the program on the diskette provided with your guide to operations, a copy of the advanced diagnostics, or a program like HTEST/HFORMAT) that can determine whether the problem is serious. Follow the instructions for the tests in your manual immediately. If the disk responds, back up the entire disk immediately, then proceed with the examination.

The second message (disk boot failure) may mean you need to recopy the operating system to the disk. The proper steps would be to boot from a DOS diskette, then execute these two steps:

```
SYS C:
COPY COMMAND.COM C:\ /V
```

The first step places another copy of the two DOS system files on the hard disk. The second step ensures that a good copy of COMMAND.COM is placed on the hard disk. If this does not work, you'll need to follow the next procedure.

The first message (bad partition table) means the master boot record is destroyed. Cylinder 0 has been damaged. The technique for

repairing a damaged cylinder 0 is similar to unformatting a disk. This technique is a "last-ditch" effort and assumes you have established and regularly maintained the BACKUP.M_U file. You may lose the files that have been created or changed since the last running of RXBAK. However, if you run RXBAK on each start-up of DOS (via the AUTOEXEC.BAT file), the damage should be minimal.

The general procedure is

- Run the Mace Utilities and use the Restore Boot Sector option.
- If the previous procedure fails, run DOS's FDISK to reestablish the master boot record; then rerun Mace to restore the BACKUP.M_U file.
- If the previous step is unsuccessful, run a physical format program on cylinder 0 only! Return to the previous step.

Once the hard disk is functional, immediately do a backup. Whatever caused the destruction of cylinder 0 could strike again. Now is the time to back up again. Do not, however, use your most recent set of backups. Use a new set instead. The failure that damaged cylinder 0 may have damaged other files. The possibility of damaging additional files is not large, but file safety is essential. You may need the previous backups to recreate damaged files.

Before following the preceding steps, shut your system down for a few seconds and then power it back up. Errant signals may have caused the hard disk to freeze. Turning the computer off and back on can clear this condition. Try to get a directory on the hard disk. If the disk does not respond, follow the procedure below:

Step 1. If DOS cannot be started from the hard disk, boot the operating system from a floppy diskette. Because the hard disk is not responding, you must start DOS from a floppy diskette.

Step 2. Run the Mace Utilities from a floppy diskette. Start the Mace program (MACE is the name of the menu program).

Step 3. Select the Restore Boot Sector option on the Utility Menu. Press special-function key F2 to enter the Utility Menu; then press F3 for Restore Boot Sector. Follow the instructions given by the program.

Step 4. If no error is received, exit the Mace Utilities and attempt to restart DOS from the hard disk. If this succeeds, you have restored your hard disk.

Step 5. If DOS did not boot from the hard disk, start DOS from a floppy diskette and run FDISK. FDISK is needed to re-create the master boot record of the disk. Directions for using FDISK are in Chapter 4. If FDISK gives you an error message, follow the next step. If FDISK does not give an error message, jump to step 8.

Step 6. Run a physical formatting program on cylinder 0 only! If FDISK has given you an error, cylinder 0 of the hard disk must be physically reformatted. However, you do not want to reformat any other cylinder. Reformatting the remainder of the disk will irrevocably destroy your data. You will need a program like HFORMAT or the IBM Advanced Diagnostics that can format a specified range of cylinders. You want to format cylinder 0 only.

If you have no errors when running the physical format program on cylinder 0, proceed to step 7. If you have an error, the disk or adapter probably has an electronic failure. You will probably need to have the drive serviced.

Step 7. Rerun FDISK. Having destroyed the master boot record with the physical formatting program, FDISK must be run to reestablish the information.

Step 8. Run the Mace Utilities from a floppy diskette and select Restore BACKUP.M_U. Run the MACE menu program from a floppy diskette and press F2 (Utility Menu) and then F5 (Restore BACKUP.M_U). This step re-creates the DOS boot record, file allocation table, and root directory. Except for files that have changed since the last run of RXBAK, all other files are now intact.

You should follow the direction in the Mace manual for recovering any partial files. You may be able to reconstruct some data files that were lost.

Formatting a Disk with FORMAT

As mentioned many times in this book, a new diskette or disk must be formatted physically and logically before useful data can be stored. FORMAT performs both tasks on a floppy diskette; FORMAT provides logical formatting only on a hard disk.

The syntax for FORMAT (IBM's version) is

*dc:pathc***FORMAT** *d: /S /V /1 /8 /4 /B*

d: is the name of the disk drive to be formatted. If the current disk is a hard disk, *d:* is mandatory. If the current disk drive is a floppy disk, *d:* is optional and, if not given, the current floppy disk drive is formatted.

The switches are

/S	Places the operating *system* on the disk (copies the appropriate BIOS, DOS, and COMMAND.COM files).
/V	Enables writing a *volume* label on the disk. (On some versions of MS-DOS, FORMAT automatically writes a volume label and */V* suppresses this writing.)
/1	Formats only *1* side of the disk, regardless of disk drive (usable only when formatting 360K diskettes).
/8	Formats *8* sectors per track only, even though normally 9 sectors per track would be created (usable only when formatting 360K diskettes).
/4	Formats a 360K diskette on a 1.2M disk drive.
/B	Reserves the proper areas on the diskette to hold the operating system. (Note: does *not* place the operating system on the diskette; usable only when formatting 360K diskettes).

The first two switches may be used on all disk drives. */S* places the proper operating system files on the disk, making the diskette bootable. When applied to a diskette, the diskette is called a *DOS diskette* (able to start DOS). */V* writes a volume label on the disk, which is shown when commands like DIR, CHKDSK, or TREE are used. The volume label also is used to confirm that the hard disk should be formatted. On some versions of MS-DOS, */V* suppresses the writing of a volume label. On these versions, you write a volume label by not giving the */V* switch.

The next four switches apply to floppy diskettes only. The switches cannot be used when formatting a high-capacity diskette (1.2M) or a microfloppy diskette (720K). The eligible diskettes are called 360K diskettes, the maximum storage possible on this type of diskette. The term is inaccurate. Two switches lower the actual storage capacity.

The first IBM PC floppy diskette drives used one side and eight sectors per track. The second generation of floppy disk drives also used eight sectors per track, but two sides. The third generation of (360K) disk drives uses nine sectors and two sides. Three switches are used by FORMAT to produce diskettes usable by the older disk

drives. */1* formats one side of the diskette only. */8* formats eight sectors only. Hence */1* is a 180K diskette; */8* is a 320K diskette; */1 /8* is a 160K diskette.

/4 allows the creation of a 360K diskette in a 1.2M (high-capacity) disk drive. Without this switch, the drive will format a 1.2M diskette.

/B reserves the proper space for the operating system, but does not place the operating system on the disk drive. The formatted diskette must have system files and COMMAND.COM copied to the diskette to make this a bootable diskette.

Formatting is irrevocably destructive with floppy diskettes; some potential for recovery exists for hard disks. See the section on unformatting the hard disk in this chapter for more information. Remember that formatting a hard disk with COMPAQ DOS V3.1 or some versions of AT&T DOS V2.11 is unrecoverably destructive.

When formatting, always give a disk drive name. This deters accidental reformatting of a hard disk. Also, be sure that the proper diskette has been inserted in the disk drive. Many times users forget to insert the correct diskette and format what was useful information. The information becomes useless after formatting.

Reformatting a nonremovable (fixed) disk is so critical an issue that IBM has added to DOS V3.1 and later versions a revalidation step. After the emphatic warning that the fixed disk is about to be formatted and all information will be lost, you are required to enter the hard disk's volume label exactly. If you fail to enter or misspell the volume label for the disk, FORMAT aborts and does not format the hard disk.

Because of the destructive nature of FORMAT, this program should be placed in a nonPATH directory or its name should be changed if casual users might accidentally invoke the program. Additional suggestions for FORMAT's placement were given in the chapter on batch files.

Moving a Hard Disk System

Chapters 1 and 3 mentioned the sensitivity of hard disks to shock. To avoid such shock to your hard disk, park the recording heads if you plan on moving the computer more than a few feet. This advice is particularly important for portable computers, which can be severely jarred during transport. The following techniques do not apply to disk drives that automatically park the recording heads. When you

power down those computers, the disk automatically retracts the recording heads from the data area into a safe, home area.

To secure the heads on some computers, you must disassemble the computer and manually turn a screw or bolt. If you mechanically prepare a hard disk for a move, you must reverse the process to use the hard disk again. This means turning whatever bolts and screws necessary to free the recording heads.

Most personal computer hard disks, however, require you to run a program to activate the park procedure. For IBM and most hard disk clones the program to park your heads is called SHIPDISK.EXE. The program for IBM machines is located on the diagnostic diskette that came with the computer. You can run the program from the menu of the diagnostic diskette, or you can copy SHIPDISK.EXE to your hard disk and run the hard disk copy. Running a hard disk copy of SHIPDISK is safe.

When you invoke SHIPDISK, the red light of the hard disk goes on and then off. A message appears telling you to turn the computer off. The heads are parked. To free the heads, simply turn the computer on. No other procedure is necessary.

Personal Computer AT owners might copy two programs to the hard disk: SHUTDOWN.EXE and SHIPDISK.EXE. SHUTDOWN graphically displays the computer's power switch and calls SHIPDISK to park the recording heads. After SHIPDISK exits, SHUTDOWN moves the switch to the off position and emits a tone, signaling that the computer should be turned off. SHUTDOWN does not add any function to the SHIPDISK process, but it can provide some humor for you and your colleagues.

Writing a Volume Label on a Disk

The volume label is an electronic name given to a diskette or hard disk. The label helps identify the removable media and is used to validate the intent to format a hard disk with FORMAT V3.1 and later versions. The volume label also appears when you use such commands as DIR, CHKDSK, and TREE.

A volume label is eleven characters long. Any character that can be in a file name can be in a volume label. Also, the space and period are legal characters in a volume label.

You have two opportunities to label a disk. The first opportunity comes when you use FORMAT. You use the */V* switch. FORMAT will ask for the volume label.

DOS V3.x users can use the LABEL command. The syntax to LABEL is

*dc:pathc***LABEL** *d:volume_label*

d: is the optional name of the disk drive whose volume label should be created or changed (if one exists). If not given, the current disk drive is used.

volume_label is the optional new volume label. If given, the new volume label is written to the disk. If omitted, LABEL will show the current volume label and then request the new label. Answer *Y* to destroy the label, *N* to retain the current volume label.

Because of the way DOS handles command lines, you may not use a space when entering a label on the command line (you give the optional *volume_label*). To enter a label with a space, invoke the command without the optional volume label. When the program asks for the label, you may enter a space as part of the name.

Summary

This chapter presents the programs and techniques for manipulating directories and disk drives. You reviewed commands for creating, changing, locating, removing, renaming, and sorting directories. The chapter covered an extensive discussion of possible problems with the physical or logical structure of a disk drive and gave remedies for many of the problems. You also learned how to format a disk, write or change volume labels, and several other disk-wide operations.

The next chapter, the final chapter in this problem-solving part, covers making the hard disk easier and more efficient to use. The major efficiency tools are RAM disks, disk cachers, and file archivers. The ease-of-use discussion includes installing data path programs (for programs hesitant to work with hierarchical directories), using more international functions of DOS, and working with copy protected programs.

Chapter 12

Convenience and Speed

This section is a potpourri. The chapter covers RAM disks, disk cachers, disk optimizers, and several other tools that can speed your operations. Included are some practical suggestions for improving system performance.

You will find discussion of general techniques and programs for making hard disk use easier, including using a PATH command for data files. The DOS commands REPLACE, GRAFTABL, and international keyboard changes are also discussed along with a review of how to manage copy-protected programs.

I make only one hardware suggestion: buy more RAM. DOS, your programs, and your data swim in this pool of RAM. This hardware suggestion is made after the discussion of the two kinds of programs that can use that RAM: RAM disks and disk cachers.

Getting Better Performance

A common desire of computer users is to "make it work faster." The following sections respond to that battle cry. There is more than one way to skin the cat of poor hard disk performance. Many available programs can improve your computer's operation without modifying

the hardware. Some simple techniques with DOS also yield better performance.

In early computing days, RAM was expensive. Most microcomputer systems in 1976 had 16K to 48K of RAM and retail prices of $450 for 16K. Today RAM is a commodity going at street prices of $36 for 256K. Although 48K was a tremendous amount of memory 10 years ago, now the 640K limit of DOS has become the major barrier. Systems with 512K are the norm. With so many systems using 512K or more, imaginative uses of RAM are possible.

Basically, moving electronic data within RAM is many times faster than moving information between RAM and an electromechanical device—the disk drive. The CPU of the system executes several high-speed instructions to dispatch information into RAM. Many more instructions are needed to dispatch the same information to a disk drive. RAM-to-RAM information moves, rated in tens of microseconds (10^{-6}), are teens to hundreds of times faster than RAM-to-disk moves, rated in milliseconds (10^{-3}). (The concept behind the DOS file buffers, discussed in Chapter 4, relies on high-speed memory-to-memory transfer.)

DOS uses RAM to hold sections of your disk in memory. Some programs use that memory as if RAM were a disk drive. Other programs are more intelligent than DOS about holding disk sections in memory and so increase performance still more. These two kinds programs are called, generically, *RAM disks* and *disk cachers*.

RAM Disks

A RAM disk is a section of RAM that is set apart by special software. This special software, called RAM-disk software, is responsible for reserving a section of RAM for its own use. The software establishes a boot record, a file allocation table (FAT), and a directory for holding information. RAM-disk software also hooks itself into DOS and handles the transfer of information between DOS and the pseudo disk. RAM operates as if the designated section of memory were a physically separate disk drive.

The RAM disk has been called a solid state disk (because of the integrated circuitry), a silicon disk (because memory circuits are silicon based), and a virtual disk (because RAM looks like a "real" disk drive). DOS and your programs happily use the RAM disk as if it were any other disk drive. You happily use the RAM disk because its throughput is several times faster than that of hard disks.

The bliss of a RAM disk ends when DOS stops. If you lose power or DOS, the information on your RAM disk is lost. If your only copy of information was on the defunct RAM disk, that information is gone. Therefore, RAM disks frequently are used to hold programs that are copied from the hard disk or a floppy diskette.

RAM disks also can hold temporary data files created by programs or unchanged data files. Generally, the RAM disk is used for frequently used program files and occasionally used data files. RAM disks are used infrequently to hold vital data files because any updates to the files are lost when the computer locks up or loses power.

Disk Cachers

The dictionary defines a *cache* (pronounced "cash") as a safe place for concealing and preserving provisions. In computers, a cache is a safe place for high-speed use of information. Caching has been a part of mainframe and minicomputers for decades. Inexpensive RAM and more powerful microprocessors now allow disk caching on microcomputers.

Disk cachers ("cash-ers") are programs that sequester a section of RAM. However, neither DOS nor your programs take notice. The functions of a disk cacher are different from those of the RAM disk. The disk cacher sits between DOS and the disk drive. When DOS requests information, the cacher intercepts the information, stores a copy in a safe place, then hands the information to DOS.

In concept, a cacher works like DOS's disk buffer. Technically, the disk buffer is a form of caching, but in practice disk-caching programs outperform DOS disk buffers. When the buffer space is exhausted, DOS recycles its buffers on a least-recently-used (LRU) basis. The buffer that has not been used for the longest time is recycled. Even though the information in that buffer may be needed soon, requests for other information force DOS to reuse the buffer.

Disk cachers use a combination scheme to determine what to keep in memory and what to discard. The disk cacher analyzes the *hits* (requests for specific sections of the disk drive). When reusing the cache, the program preserves the sections with the most hits, the most frequently used sections, and discards the least frequently used information.

Given absolutely random disk access (files and data are randomly scattered on the disk), the least frequently used system works best. Given intense operations of several disk sectors within a short period,

the least recently used system works best. Disk cachers, by dynamically analyzing your use of the disk, strike a balance between the two schemes. The result is much higher disk performance than DOS buffers can provide.

Some cachers offer other benefits. The original concept behind disk buffers was to minimize the number of disk reads. Frequently, a program requests information that is smaller than a disk sector. By reading a full sector at a time, DOS eliminates the need to reread partial sectors. When the next part of the sector is needed, DOS parcels the information to the program from the hidden file buffers.

In addition, some disk cachers go one step farther than DOS and read a full track of the disk at one time. Instead of individually reading the 9, 15, or 17 sectors on a track, the disk cacher reads all sectors at one time. This full-track-read approach dramatically reduces the time needed to load sequential files, such as programs and their auxiliary files, word-processing documents, and spreadsheets.

Disk cachers also buffer file writing. Cachers determine which sectors of a file should be changed and so eliminate unnecessary writes. For example, suppose that your program will write four sectors to the disk, but only sectors one and three have been changed. The disk cacher writes sectors one and three only. Therefore, 50 percent of the writing of information to the disk has been avoided.

Because writing information to the disk takes longer than reading information, eliminating even a small percentage of disk writes yields significant improvement. Some disk cachers also increase disk performance by doing full-track writes, writing all sectors of a track at once rather than each track individually.

Except in rare cases, disk cachers outperform DOS disk buffers. The increased performance with a disk cacher is exceptional.

Contrasting RAM Disks and Disk Cachers

Newcomers often ask the following questions about RAM-disk software and disk-cacher programs:

Do RAM disks and disk cachers do the same thing?

What are the risks of both?

Should you use a RAM disk, a disk cacher, both programs, or neither?

These questions are valid and are addressed in this and the following sections.

A RAM disk and a disk cacher are not the same; the two are similar but different. Both work from RAM, and both improve disk operations, but each improves different disk operations and has different purposes.

RAM disks reside in volatile RAM. The disk and its contents disappear when power is lost or DOS is restarted. Existing files must be copied from a real disk to the RAM disk before they can be used. If a file on the RAM disk is altered, you must perform the additional step of copying the file to a physical disk to preserve the alterations.

A disk cacher works with the file residing on the physical disk drive. The disk cacher anticipates your needs and keeps key sections of files in memory. If the cache size is sufficient, complete files are held in the disk cache. Except for brief periods when the cacher assembles information to be written to the disk, the copy of the file on disk is current. When power is lost or DOS is stopped, the cacher loses its analysis of your disk use, but the underlying disk files are intact.

Basically, a RAM disk is used for manipulating complete files, and you must manually copy files to and from the RAM disk. A disk cacher can manipulate parts of files, and automatically handles the copying between the disk drive and the cache.

The risks of using a RAM disk and a disk cacher when reading information are identical: none. No risk is involved when reading information from a file. The file remains intact on the physical disk; only an easily reproducible electronic copy can be lost.

When altering files, however, the potential for losing the revision is greater with a RAM disk than with a disk cacher. With a RAM disk, the file's alterations are held in RAM. Until copied to the physical disk, the revision in memory is at risk. Any disruption to DOS causes the loss of this information.

A disk cacher holds the revision in memory for a short time, small fractions of a second to several seconds. Because the delay in updating the file is so short, the chance of losing information is very small. Except when the cacher delays updating for a second or more, the chance of losing information is no different between disk cachers and DOS disk buffers.

Choosing between a RAM Disk and a Disk Cacher

Based on their operational benefits and risks, you can probably use both a RAM disk and a disk cacher. However, they have different ranges of best uses.

The RAM Disk

A RAM disk is best used with two types of files: unchanged files and transient files. The term *unchanged files* is almost self-explanatory: files that are not altered and are only read from the disk. Because all files are replaced at some time (programs updated or data files changed once every two weeks to two months), *unchanged* is relative to time. Unchanged files include programs and their auxiliary files and data files that are rarely altered. These files are best used on a RAM disk because little risk is involved.

While unchanged files have a long life, transient files are created, used, and destroyed within a period of seconds or minutes. Many programs create temporary files because the information they process cannot be kept in RAM. The disk is used as a short-term storage area. Rather than using the slower physical disk storage, using the supercharged RAM disk can greatly improve the program's throughput. Frequent temporary file users are word processors and program language compilers. DOS also creates temporary files when you use piping (redirect the output of one program as the input to a second program).

Another popular use for the RAM disk is to hold the most frequently used DOS programs or utilities so that the frequent commands load faster. You also can use the RAM disk to hold frequently used unchanged data files; for example, a spelling checker's dictionary files. Any files that are loaded from the disk more than once in a day can be placed on a RAM disk.

Still another use for a RAM disk is for copying files between two floppy diskettes when the computer has only one floppy disk drive. After the destination diskette is formatted, copy the files to the RAM disk, change diskettes, and then copy the files to the destination diskette. With the high speed of the RAM disk, performing the two-step operation takes less time than using DISKCOPY or using the hard disk for the temporary copies.

Your changing data files can be placed on and used from a RAM disk. The improvement in speed makes this practice justifiable, but one caution must be constantly observed:

> The alterations are not permanent until the file is copied from the RAM disk to a real disk.

At any given time, the chance of a DOS disruption (power outage, computer lockup, system reset, and so on) is low, say two thousandths of a percent every second. In time, however, the odds compound. The longer the delay before copying the files to a real disk, the greater the risk of losing the changed files. Microcomputers are not invincible. Any disruption will be fatal. You must practice the discipline of regularly copying the files to permanent storage.

How often should you copy the altered file from the RAM disk? Frequently and often. Actually, I liken the process to gambling: gamble only with files you can afford to lose. If you can re-create the data in a few minutes, you can copy the file to a real disk after you leave the program that uses the file. You even can delay until the time you shut off the computer or leave it for the day.

If the data on the RAM disk cannot be reconstructed easily, copying the changed files every 15 minutes to an hour should suffice. If you cannot possibly reconstruct the data, don't put it on a RAM disk!

The Disk Cacher

The best time to use a disk cacher is any time. Because the cacher sits between DOS and your disks, the cacher benefits any disk activity. The biggest benefits come during random disk reading and writing within a set of files. Usually, less benefit occurs when sequentially reading files from the disk. But because disk cachers frequently hold the "hot" DOS housekeeping areas of directories and the file allocation table, you often see some improvement when reading sequential files. Even for sequential files, disk cachers with full-track reading capability improve performance. For example, I have seen the time to load a 1-2-3 worksheet decline from 12 seconds without the cacher to 4 seconds with the cacher.

On some occasions disk caching offers no benefit. A common example is using DISKCOPY. Because DISKCOPY reads and writes one disk track at a time, disk cachers cannot help. Even if the same disk is recopied, DISKCOPY automatically rereads the source diskette again. Although cachers cannot help DISKCOPY, cachers do improve over 95 percent of other disk operations.

Competition for RAM

Undoubtedly, you've reached the conclusion that both RAM disks and disk cachers are desirable. You also probably have guessed that the price for this increased performance is the use of RAM. Unfortunately, DOS imposes the rude limitation of 640K for RAM. RAM disks and disk cachers vie for the same RAM used by DOS, your programs, and their data. Other resident programs, such as SideKick or ProKey, also may be in competition for your limited RAM.

While 640K of RAM seems like paradise, many personal computer owners find that 640K is becoming a critical limitation. Although DOS or the largest spreadsheets rarely use 640K, your programs and resident programs create the irresistible force that meets the immovable wall of the DOS 640K limitation.

Generally, robbing Peter (your programs) to pay Paul (your RAM disks and disk cachers) works poorly. Your RAM disks and disk cachers do not create productive work; your applications programs do. Therefore, you must strike a balance that increases disk performance while maintaining enough RAM to satisfy your programs.

Once installed, most RAM-disk software cannot be dynamically uninstalled. You cannot use a DOS command to free for your programs the memory used by the RAM disk. Basically, once you start with the RAM disk, you finish with it. The only way to free the memory is to remove the appropriate entry from the CONFIG.SYS or AUTOEXEC.BAT file and restart DOS.

A few program-based RAM disks can be uninstalled at the command line. However, if other memory-resident utilities are used with the RAM disk (such as Sidekick or DOS's PRINT and MODE commands), the memory cannot be freed successfully. Resident programs currently demand a last-installed, first-uninstalled approach. If you uninstall programs in the wrong order, the remaining programs and DOS act erratically. Most disk cachers can be uninstalled, but under the same caveats as program-based RAM disks. If you remove programs out of order, your computer acts erratically.

Because of limited RAM space, you may be forced to choose between a RAM disk and a disk cacher. Your best choice is the disk cacher. The RAM disk helps when you are working on complete files. The disk cacher can help with any disk operation, not just complete files. The RAM space stolen from your programs is better spent on the disk cacher.

How much memory can be devoted to the disk cacher? The program itself occupies 20K to 30K. The size of the cache can vary

from 5K to hundreds of K. Although a 5K cache helps, a 30K to 100K cache is more practical. Larger cache sizes (100K to 1000K) do improve performance but may not be practical. The typical balancing point is 70K to 100K.

Lack of memory is also an impediment to large RAM disks. Most people use 60K to 360K RAM-disk sizes. Although RAM disks can be as large as your RAM size minus DOS's size minus 64K for your programs, a RAM disk this large is impractical.

Solving Problems with Expanded and Extended Memory

The 640K limitation of DOS has frustrated many users. However, the immovable wall is yielding to the irresistible force. There are two immediate and one upcoming solutions to RAM cram: expanded memory, extended memory, and multi-tasking DOS. The first two solutions are ingenious and exciting.

The native central processing unit of the IBM PC is the Intel 8088, cousin to the 8086. Both CPUs are 16-bit processors that can directly address 1M of memory. Some space is devoted to read-only memory (ROM); some is devoted to RAM used on various adapters, such as the video display adapters. 640K of space is devoted to nondedicated user memory—the memory DOS and your programs use. That is the limit of the 8088 and 8086 processors and IBM's design.

If you carefully peruse the technical manuals for the personal computer family, you'll note that IBM designed gaps in the memory. Enter the concept of *expanded memory*. An unplugged 64K of space exists above the 640K point. Intel, Lotus, and Microsoft combined their knowledge to place a memory board in this 64K gap. The board uses a multiplexing scheme. Basically, the RAM on the board is organized into 16K groups, each called a *page*. Circuitry on the expansion board both hides and exposes pages of memory to the view of the 8088/8086. With the proper software control to enable and disable pages of RAM, up to 8 megabytes of information can be crammed through this 64K gap.

The original expanded memory concept has two names: *EMS* memory, for expanded memory specification, and *LIM* memory, for Lotus, Intel, and Microsoft. Expanded memory is an ingenious attempt to overcome the 640K limitations of DOS and the 8088 and 8086

CPUs. For example, 1-2-3 Release 2 users can create multimegabyte spreadsheets.

Expanded memory has a problem: most programs cannot operate from expanded memory. The 640K limitation of DOS continues to limit the size of programs or the number of programs that can reside in the computer at one time. Programs can use the expanded memory for their data, however. This fact means that RAM-disk programs and disk cachers can put their disks or cachers in expanded memory. With the proper amount of expanded memory, your RAM disks and disk caches can be megabytes in size. Although many programs do not directly use expanded memory, you can justify the cost of expanded memory by using RAM disks and disk cachers. Your disk throughput can be dramatically improved without robbing your programs of work space.

Expanded memory has a second flavor. AST Research and Quadram, two strong players in the add-on market, announced a superset of EMS called Extended Expanded Memory Specification, *EEMS*. Also known as the AST/Quadram specification, EEMS is not fully compatible with EMS. However, AST and Quadram are shifting their EEMS boards to work like EMS boards also.

If these terms are dizzying, the next set further clouds the issue. From the depths of the engineering of Intel microprocessors comes *extended memory*. Extended memory is memory beyond the 1M point. The 8088 and 8086 address 1M of memory. The Intel 80286, CPU of the IBM Personal Computer AT class of computers, addresses 16M of real memory. The 80386, the next generation CPU used in the innovative COMPAQ 386 DeskPro, addresses 2G (two gigabytes or two billion bytes) of real memory. Both machines use extended memory.

The 286 and 386 have two different modes of operation. In the *real* mode, the two processors emulate an 8088/8086, including respecting the 1M memory limit. In *protected* mode the machines can address all potential memory, not just the first 1M. Often, therefore, extended memory is called *protected memory*.

Obviously, expanded and extended memory are not the same. Expanded memory can be used on almost all 8088/8086/286/386-based computers; extended memory can be used only on 286/386-based computers.

If you are looking for a memory board, which should you use choose? Owners of the PC, PC XT, and similar computers can choose only expanded memory boards. Extended memory is not possible. Owners of the 286 and 386 have a choice. Because paging memory

is faster than entering and leaving protected mode, expanded memory is faster than extended memory. However, future versions of DOS, which will allow true multi-tasking capabilities, will use extended memory.

Expanded memory is today's solution; extended memory is tomorrow's. At a slightly higher price, you can get the best of both worlds. Several memory-board makers are selling boards that work in both modes. You can work in the faster expanded memory mode or work in the future's extended memory. If you are unsure of the longevity of your system, the dual expanded/extended board is the better choice.

Using RAM-Disk Software

RAM-disk software abounds. There are two types of RAM disk software: the type used as a device driver and the type run after DOS has started.

If you have DOS V3.0 or later, you have on your system disk a file called VDISK.SYS. The name translates roughly to "virtual disk system software," with the latter two words meaning "device driver." Some RAM disks, such as VDISK, are installed through the DEVICE= command of the CONFIG.SYS file. (The following section covers how to install VDISK.)

The other type of RAM disk software is a program that can be run from the DOS command line. The command line also can be used in an AUTOEXEC.BAT file. Many RAM memory boards come with RAM-disk software bundled in. The executing instructions are peculiar to the memory board you purchase. For example, the AST SixPack memory board has a RAM-disk program called SUPERDRV.COM. To create a RAM disk for this board, I invoke the command

SUPERDRV D:/M=100

This line creates a 100K drive called D:.

Depending on your computer and type of RAM-disk software, you may need to reset the configuration switches to tell the computer you have another disk drive. Check the program's documentation to learn whether such changes are required. Obviously, RAM-disk software that requires no switch resetting is preferable to software that requires the setting.

Not all RAM disks work in expanded or extended memory. Usually the memory board manufacturer provides the RAM-disk software that works with your board.

Using VDISK for a RAM Disk

Owners of DOS V3.0 and later can use the VDISK.SYS device driver. Only one line is added to the CONFIG.SYS file. The general form of the line is

DEVICE = *d:path***VDISK.SYS** *comment bbb comment sss comment ddd /E:max*

The optional *d:path*\ is the disk drive and path name to VDISK.SYS. Unless VDISK.SYS is in the root directory of the start-up disk drive, you must give the disk drive name and path.

The four options for VDISK are the following:

bbb	The size (in K) of the RAM disk
sss	The size of the RAM disk sectors
ddd	The number of directory entries (files) for the RAM disk
/E:max	The option for extended memory use

The value of *bbb* can range from one to the maximum amount of memory in the computer minus 64K. If you don't give a size or give a size larger than your computer's memory, VDISK defaults to a 64K size. If you specify a size that leaves less than 64K for your programs, VDISK adjusts its size downward to leave at least 64K of RAM free.

The *sss* is the size of the disk sectors for VDISK. You can specify 128, 256, or 512 bytes. If you omit the sector size or give some other value, VIDSK defaults to 128-byte sectors.

For *ddd* you enter the number of directory entries for the RAM disk. As each file uses one directory entry, *ddd* specifies the maximum number of files the disk can hold, *minus one.* One directory entry is automatically used for a volume label. Note that the range of *ddd* is from 2 to 512, and the default value is 64.

The optional switch */E* is for extended memory. Only 286- and 386-based systems, such as the Personal Computer AT and XT/286 and the COMPAQ DeskPro 386, can use extended memory. The */E* switch varies between DOS V3 revisions. V3.0 uses */E* and V3.1 and V3.2 use */E:max*, in which *max* is the maximum number of sectors to transfer at one time from the RAM disk. Certain programs, particularly communications programs, may drop characters if the

computer keeps its attention on the RAM disk for too long. If you notice problems, set *max* to 7 or less. Otherwise, omit the *:max* or specify it as */E:8*.

If you have an add-on memory board that provides more than 1M of memory, you can use the /E switch. Otherwise, you cannot. VDISK issues an error message when the program finds that extended memory is not available and VDISK does not install itself.

The optional *comments*, available only in DOS V3.2, merely allow you to make the command more readable. The comments can contain any ASCII characters in the range 20 through 126 except the slash (/)—basically, the alphanumeric and some punctuation characters. Here is a simple example of a VDISK command line with comments:

DEVICE = C:\BIN\DRIVERS\VDISK.SYS BUFFER SIZE=128 SECTOR SIZE=512 DIRECTORY ENTRIES=16

The characters of the comment are ignored by DOS.

The simplest form for VDISK specifies only the size of the disk. Assuming VDISK.SYS is in a subdirectory called C:\BIN\DRIVERS, use the CONFIG.SYS line

DEVICE = C:\BIN\DRIVERS\VDISK.SYS 160

This statement establishes a 160K RAM disk. VDISK defaults to a sector size of 128 and uses 64 directory entries. The following example establishes a 1M VDISK in the extended memory of a 286- or 386-based computer, assuming the extended memory is present:

DEVICE = C:\BIN\DRIVERS\VDISK.SYS 1000 /E

As indicated by the examples, I recommend that VDISK.SYS and other device drivers be placed in the \BIN\DRIVERS or \DOS\DRIVERS subdirectory.

The essential parameter to specify is the size of the RAM disk. The size must be determined by the amount of RAM in your system, the programs you use (including disk cachers), and the convenience of the RAM disk. Owners of the 286 or 386 should remember that the RAM disk can reside principally in extended memory, if available. Owners of expanded memory must use a different type of RAM disk.

Using Disk Cachers

Interestingly, disk cachers produce better results than DOS and its file buffers. Given the choice between allocating memory to buffers or memory to the cache, the cache is the better choice. However, disk cachers have more overhead. The caching program itself needs some RAM. When free memory is limited, you may need to revert to operations without the disk cacher. If you plan on performing some tasks without the disk cacher, you should leave the BUFFERS line in the CONFIG.SYS file or reboot from a diskette whose CONFIG.SYS file reflects the modified BUFFERS line. Some DOS buffers are better than no file buffers when a disk cacher is not active.

The field of PC disk cachers is growing. You now have a choice. Two better-known disk cachers are Lightning (not to be confused with a Borland product called Turbo Lightning) and Flash. The similar names represent what their manufacturers think of each program's advantages. Depending on your particular application, the manufacturers are right. Some applications do run like a Flash of Lightning. Because the two programs are functionally equivalent, the similarities are described first, then the differences.

Flash and Lightning—Similarities

As already mentioned, a disk cacher loads itself into memory and reserves some RAM for the disk cache. You specify exactly how much RAM you want for the cache (above the 20K to 40K for the program itself) when invoking the program. The cache size can be changed by "unloading" the program and restarting it with a new cache size. The disk cacher must be the last installed resident program for this procedure to work successfully.

Caching can be enabled or disabled for selected disk drives. Selective use of caching is helpful when the cache size is insufficient to handle your program files and your data files. You place the data files on a separate disk and direct the cacher to work with this data disk only, allowing the cacher to ignore buffering the program files. (An alternative to this approach for Flash is given in the next section.)

Both programs are easy to install and use. The ease of use does not sacrifice flexibility. Advanced options are available, but they do not hinder the newcomer who does not use the features.

Each program comes with a demonstration program, which showcases the program's advantages. You watch the short program signal the screen each time a record is read randomly from a disk file. The demo program runs with and without the disk cache and quickly and dramatically demonstrates performance with and without the cache.

Both programs incorporate an option to get quantitative feedback on the improvement in your disk access situation. Lightning's status screen is shown in figure 12.1 and Flash's is shown in figure 12.2. The statistics report total disk reads and writes, the number of read/write operations the cacher has "saved" (Lightning calls them "hits"), and the percentage of disk operations saved. The statistics verify possibly vague feelings that disk operations are faster and help quantify the savings. The statistical counters can be reset to zero so that you can "clock" a particular application or set of events.

Both programs provide an option to write-protect selected disk drives, similar to covering the write-protect notch on a floppy diskette except that the protection is provided through software. The hard disk can be protected also—a feature rarely available in other programs. Lightning's documentation warns that some backup programs write directly to the disk controller, thus bypassing Lightning completely and negating its write protection.

```
LIGHTNING Version 4.20

Copyright (C) 1985 Personal Computer Support
Group, Inc. All Rights Reserved
"LIGHTNING" is a trademark of PCSG, Inc.

Type L HELP for options summary

LIGHTNING is speeding up drives A B C F G
Total RAM in use is  60 Kbytes

Since LIGHTNING was first loaded or cleared, you have had:

  61 disk requests
  28 LIGHTNING reads
  12 regular reads
   0 LIGHTNING writes
   0 regular writes

LIGHTNING has reduced disk accesses by 45%

A:\ -->
```

Fig. 12.1. *Lightning's status screen.*

```
A:\ -->flash /s
```

```
FLASH
STATS

Memory:100 K-C  Priority:L  Active:Y

Turned OFF:
Write-Pro :
Hi-Pri Used:00001        Free:00070

            Reads        Writes
Real        0000057      0000006
Saved       0000149      0000001
            -------      -------
Total       0000206      0000007

Saved          072%         014%
```

```
A:\ -->
```

Fig. 12.2. *Flash's status screen.*

The documentation for both programs emphasizes that using the CONFIG.SYS BUFFERS= command neither hurts nor helps. Contrary to the advice given in Chapter 4, you can safely reduce or eliminate the BUFFERS command when you are using disk cachers. Performance will not decrease.

Flash and Lightning—Differences

Although both programs work similarly, they have some differences in features and operations. Lightning comes in copy-protected and nonprotected forms. Flash comes in a single nonprotected version. If you customarily start DOS from a hard disk, your only choice is a nonprotected version. When comparing prices, compare prices of nonprotected versions.

With software, the saying "You can tell a book by its cover," or more precisely, its documentation, is often *not* true. Lightning's documentation appears more professional looking, but both Flash and Lightning's documentation concisely represent their features with enough examples to make things clear. Lightning's documentation is briefer than Flash's because Flash has more features to cover. Obviously, the Lightning developers wanted to offer a useful set of features for most PC users, and the Flash developers enhanced their software with some advanced features for more sophisticated users.

At start-up, Lightning assumes a default cache size (64K) if a size is not specified. Flash assumes nothing and displays a help screen if you do not give a cache size. Flash's basic in-memory program size (exclusive of the cache) is larger than Lightning's. Flash is about 16K and Lightning is about 6K. The larger size of Flash is easily excused by Flash's larger range of features.

Because the disk cacher market is highly competitive, a jump in features by one program is followed by the catch-up of the others. Currently, Flash (V5.0 at the time of this writing) has more features than Lightning. Flash can use expanded (EMS) memory and extended (protected/286-386) memory, thereby helping to overcome the 640K main memory limit of DOS. Lightning supports only expanded memory. Flash's cache size can be larger than Lightning's. Flash handles up to 1.7M of expanded or extended memory. Lightning's expanded memory limit is 1.5M. Most users will not approach either program's upper limit.

For handling larger caches, Flash is more efficient than Lightning. Both programs use some main memory for housekeeping when handling expanded or extended memory caches. Lightning uses about 50K of main memory to cache 500K of files, and Flash uses less than 20K of main memory for the 500K cache. When using large caches, Flash takes less main memory than Lightning.

Flash has a "high priority" mode, in which you can unconditionally lock files or disk sections into the cache. Thus, selected applications programs and their data files can be preserved in the cache. If you exit from the application and come back later, the same files are in the cache. Your disk operations with these files are further speeded. If you do not use the high priority mode, Flash determines by your use of files what sections to lock into the cache.

Flash has an option to save its housekeeping information to disk. When restarting Flash later, you can restore the appropriate information. With this feature, Flash remembers your disk usage and reloads and maintains the frequently used disk areas in memory *even after the computer has been turned off and on again.* You can also save high-priority information for an application, release the cache for normal use, and restore the high-priority information when returning to the application.

Flash has one feature not duplicated by several cachers: Flash can buffer writing to a floppy diskette. Currently, Lightning buffers only reading from a floppy diskette. Flash is one of the first cachers to perform full-disk-track reads and writes, a feature not duplicated yet by Lightning. The full-track buffering takes another 17K of memory,

boosting Flash's average main memory use to about 36K. The main memory lost to Flash is well spent and even the worst possible files to cache, sequential files read only once, benefit greatly.

Lightning is the cacher of choice for those with limited memory; Flash is the choice for users who have extended or expanded memory or use floppy diskettes more than occasionally. Flash is richer in features and actually takes less memory when using larger caches. The full-track read/write capability of Flash benefits virtually any disk operation.

Managing Disk Storage

Although 20, 30, or even 70 megabytes of disk storage seem spacious, the best definition of mass storage is "never massive enough." Eventually, you will find yourself cramped for space. This section describes some techniques and programs that help you cope with "unlimited" storage space that suddenly becomes very limited. This section also will help users of public-domain or shareware programs. Many of these programs are distributed in "condensed" form and require some processing before the program is used. This section discusses how to prepare these programs for use.

Getting More Disk Space

As you recall, a one-byte file on your disk actually may occupy 1K, 2K, or even 4K of disk space because DOS allocates space by clusters. The worst case is the 10M disk drive, which allocates space in 4K chunks. You are penalized severely if you use many small files.

One technique to reduce the allocation penalty is to place several short, infrequently used files in one larger file, which is called a *library* or an *archive*. The single library file takes less allocated disk space than many short files. Another technique uses sophisticated algorithms to compress infrequently used files so that each file takes less disk storage. These programs are called *compressors*. Some shareware and public-domain programs combine both techniques.

The motivation for creating libraries and compressors came from users of computer bulletin boards. Because communication speeds were slow, computerists looked for ways to shorten the transmission time of files. The most cost-effective way was to shorten the file or files.

If you look at lists of files on these bulletin boards, you will notice file names with strange extensions—ARC, LBR, LQR, CQM, DQC, and so on. Each of these programs has undergone some combining or compressing. If you understand the extension, you can tell which technique has been used.

The granddaddy compression program is SQZ, for squeeze. The program first appeared in the CP/M world as a means to compress a file so that it takes less storage on the bulletin board and transmits faster. When squeezing a file, the program changes the second letter of the extension to Q. For example, CQM files are COM (command files) that have been squeezed, and DQC files are squeezed DOC (document files). The amount of compression varies. Typically, the compressed file takes 65 percent of the space of the original file—a 35 percent savings.

Because memory space inside the computer is limited, SQZ's sole purpose is to squeeze files. A complementary program, appropriately named USQ, "unsqueezes" the files. The two programs are old CP/M favorites that have made their way into MS-DOS.

The computer bulletin-board community also faced another problem. Most programs occupied more than one file. Minimally, one file held the program, and a second file held the documentation. The person wanting the program needed to find and download both files. The computer community responded to this problem with *librarians*. These programs combine several files into one larger file so that only one file is downloaded. An advantage for hard disk owners is that the single file takes less disk space than the sum of the individual files.

In DOS, LU.EXE and LAR.EXE take many files and combine them into a library or extract files from an established library. Both programs use the extension LBR for library file and embed the name of each file stored into the library. When a file is extracted from the library, LU or LAR restore the file under its original name.

You have probably guessed that LQR files are squeezed library files. LU and LAR do not compress files; they simply store many files in a single file. SQU works on one file at a time. The technique takes two steps: combining the files using LU or LAR then squeezing the resultant library file. In this way, the best features of file compression and file libraries are obtained.

A recent generation of programs, called *archivers*, eliminates the two-step process. They use more sophisticated techniques to combine and compress files than SQU does. The extension of the archive files is ARC.

Because some shareware and public-domain programs are still in squeezed or library form, I discuss all three program sets. Use of the first two program types is declining, so these programs are explained briefly. Later in this section, I discuss the archive programs at more length.

Squeezing and Unsqueezing Programs

SQZ is the squeezer program. CP/M calls the unqueezer USQ. MS-DOS has adopted two unsqueezing programs: ALUSQ.EXE (all unsqueeze) and NUSQ.EXE (new unsqueeze). Because the programs have one purpose, the command lines are simple. The syntax is as follows (NUSQ is the example, but ALUSQ uses the same basic syntax):

*dc:pathc***NUSQ filename.ext** *d:*

The *filename* is the name of the file to unsqueeze. The optional disk drive name (*d:*) directs NUSQ to place the unsqueezed files on a separate disk drive, a useful practice when disk space is low. For example, to unsqueeze a library file called TRICKS.LQR the command is

NUSQ TRICKS.LQR

This statement produces the TRICKS.LBR file on the default disk.

Some versions of the unsqueeze programs do not accept a disk drive or path name for the files to unsqueeze (as shown), but some do accept wild cards in the file name. The command to unsqueeze every LQR file on a disk is

NUSQ *.LQR

The unsqueeze utility reports on its progress as it goes, naming the file being operated on and the resultant unsqueezed file

```
TRICKS. LQR -> TRICKS. LBR
TICTACK. LQR -> TICTACK. LBR
```

Most variants accept more than one file name on the command line, such as

NUSQ TRICKS.LQR TICTACK.LQR EZ.EQE

Note that squeezed library files (LQR files) must be unsqueezed before they are manipulated. When you are placing unsqueezing programs on your disk, I suggest putting them in the \UTILS or \BIN directory.

Using Library Programs

The programs LAR.EXE or LU.EXE (or perhaps LUxx.EXE, where xx is a version number) are used to extract files from a library file (LBR) or to build a library. Because the principal use of these librarians is extraction, I cover mainly the extraction commands. The syntax for both programs is close (I show LU but LAR's basic syntax is identical). The optional switches may vary. The command syntax is

*dc:pathc***LU** *-tlaeudrs d:path\library_name d:path\filename.ext*

LU follows UNIX conventions for switches; the switch appears between the command name and the first file name. The switches are preceded by a minus sign (-). You may have more than one switch after a single minus sign (*-tlaeudrs* in the syntax line). The switches are given in table 12.1.

Table 12.1
Switches Used in LU Program

Switch	*Function*
-t	list *the* files in the library
-l	*list* the files in the library (-t and -l are synonyms)
-a	extract *all* files from the library
-e	*extract* the named file(s) from library
-u	add the file(s) to the library (*update*); also create a new library if the named library does not exist
-d	*delete* the named file(s) from the library
-r	*reorganize* the files within the library
-s	date/time *stamp* each file within the library with the library file's date and time

The first file name is the name of the library file, *library_name*. No extension is given because the extension LBR is assumed. Following the switches and *library_name* are the file names; each name has an optional drive and path name. More than one *filename* can be given and wild cards are allowed.

The principle switches are *l* or *t*, *e*, and *a*. Using a hypothetical library called TEST.LBR, the command to list the names of all files in the library is

LU -T TEST

To extract all the files in the library, you type

LU -A TEST

To extract selected files, you type each file name separately, as in

LU -E TEST TEST.TXT TEST.COM

To build a new library called TEST1.LBR with the files TEST.TXT and TEST.COM, the command is

LU -U TEST1 TEST.TXT TEST.COM

Although ARC is the preferred method of joining files into a file, you may need LU or LAR to unpack files obtained from a bulletin board. I suggest that LU or LAR be kept in the \UTILS or \BIN subdirectory. Remember that a squeezed library file (LQR) must be unsqueezed before the files can be extracted from the library.

Archiving Programs

Competition hits all sectors of the computer industry. The first major archival program was ARC.EXE. This program is a combination compressor and librarian. You'll probably see the program listed as ARCxxx.EXE. The xxx represents the version number. (ARC V5.12 is current as of this writing.) Because ARC typifies the combination compressor-librarians, it is used in this discussion. Two other major entries are ARCA/ARCE and PKARCA/PKARCX. These two sets of programs are described at the end of the discussion.

ARC does not just squeeze files. ARC reads the files and determines whether they should be simply stored (the compression housekeeping information would be bigger than the file) or compressed by using one of three routines. The squeeze routine is based on the work by David Albert Huffman for constructing a binary search table. ARC uses two other routines, one based on the Ziv-Lempel algorithm, which yield more compact files. The result is improved disk space savings.

If you forget all of ARC's functions, you simply type

ARC

After some introductory information (a brief commercial), you get a list of commands. Most librarians and archivers give their help message this way. The full syntax is

*dc:path***ARC** *amufdxerplvtc[bswn][g<password>]*
d:path\archive_name d:path\filename.ext . . .

I'll discuss ARC's somewhat intimidating switches in the following paragraphs. For now, note that LU's and ARC's basic syntaxes are similar. The first named file is the archive file with an assumed extension of ARC. More than one file name can precede the archive name, and wild-card characters are allowed in each file name.

ARC Switches

A list of ARC's switches is given in table 12.2. Note that no switch character precedes the switches (no slash or minus sign is used). To help explain ARC switches, I classify the switches into three sets.

ARC has a basic set of switches, shown between ARC and the set of switches enclosed in brackets. These switches are mutually exclusive; you can give only one of these switches on the command line. Any combination of switches from the second and third sets of switches (the switches within the brackets) can be given with any switch of the first set. The third set is the password switch for encrypting and decrypting files. The *g* switch, which should be the last switch given, is immediately followed by the password. If using this switch, do *not* forget the password. Your file could be hopelessly unrecoverable!

Reviewing the major switches in order of complexity, *l* and *v* list the files in the archive. The *l* gives the file name, the original (noncompressed) length, and the date. The *v* includes more statistics; for example, the compressed file size and the efficiency of the compression.

The *a*, *f*, *m*, and *u* switches add files to the archive. The *a* is the vanilla add switch. The *m* adds the file and deletes the original file from the disk after the file is successfully added (a move). The *f* and *u* switches add or add and replace files in the archive. Both switches cause ARC to examine the dates and times of the archived files. If a file in the archive is older than the file on the disk, ARC deletes the old file in the archive and adds the new file. The *f* replaces files that already exist in the archive. The *u* (update) works like *f* but also adds files that do not exist in the archive. The *d* is the opposite of the *a* switch and removes a file from the archive (delete).

The *e* and *x* are identical. ARC extracts a copy of the file, but a copy of the file remains in the archive. ARC does not store the original location of a file. Therefore, you can specify a disk drive and path name with the name of the file to extract. ARC will place the archive file on the specified disk drive/subdirectory.

Table 12.2
ARC Switches

Switch	*Function*
First set (only one may be given at a time)	
a	*add* file(s) to the archive
m	*move* (copy then erase) the file(s) to the archive
u	*update* the file(s) in the archive
f	*freshen* file(s) in the archive, add or add and replace files
d	*delete* file(s) from the archive
e	*extract* file(s) from the archive
x	*extract* file(s) from the archive (e and x are synonyms)
r	*run* (execute) file(s) directly from the archive
p	*print*, or copy the file(s) from the archive to standard output (to the screen)
l	*list* the file(s) in the archive
v	*verbose* listing (includes more statistics) of the file(s) in the archive
t	*test* the integrity of the archive
c	*convert* an archive file from a previous version to a new version
Second set (one or more may be given at a time)	
b	do not delete (retain) the *backup* copy of the archive
s	*store* the file only, suppress compression
w	suppress *warning* messages
n	suppress *notes* and comments
Third set	
g	encrypt/decrypt the file in the archive (*garble* the file)

The *p* and *r* switches are improvements over LU and LAR. You can print the contents of a file to the screen using the *p* switch. This method skips the step of extracting the file and then displaying it.

The displayed version can be redirected to the printer using >**PRN**. The *r* switch runs a program from the archive. In both cases, a temporary file is created. After the program prints or runs the file, the temporary file is removed.

The *t* tests whether the library is intact. The *c* converts archives created under previous versions to new packing methods. This switch is useful to update archive files when new versions of the program are released.

The next four switches are suppressors. When manipulating an archive, ARC creates a new copy of the archive and deletes the old copy. You stop ARC from deleting the old copy (which becomes the backup copy) with the *b* switch. The *s* stops ARC from compressing a file. The *w* suppresses warning messages and *n* suppresses notes and comments.

Files in the archive may be encrypted. The *g* switch (which must be the last switch on the line) is immediately followed by a password. ARC uses the password and the file's date and time to scramble the file. The protection is not infallible, and if you lose the password, you may need days or weeks to recover the file (if at all).

ARC Examples and Suggestions

Although so many switches seem intimidating, the ones you normally use are *x*, *a* (or *m*), and *l* (or *v*). As you gain familiarity with the program, the other switches will become natural to you.

To build an archive file named TAXES from several WK1 files in the current directory, you type

ARC A TAXES *.WK1

If TAXES.ARC does not exist, ARC creates the file before adding the worksheets.

To extract all files from TAXES.ARC, the command is

ARC X TAXES

When you are extracting files, specifying no file is the same as specifying *.* as the name of the files to extract. To see a list of files within TAXES, you use either

ARC L TAXES

or

ARC V TAXES

One drawback of ARC is the time needed to compress a file. The length of time depends on the length of the file and may take up to a few minutes for several 100K files. Extraction from the archive takes considerably less time. Another down side is the temporary file(s) ARC creates. Using ARC on an extremely crowded hard disk or on floppy diskettes can be difficult.

The best use for ARC is to accumulate a set of small files into one file. Programmers can keep all files related to a project in one archive. Word-processing users may place infrequently used documents in an archive. Spreadsheet users can archive infrequently used worksheets. Because ARC can reduce the size of a file by as much as 50 percent, much space is saved. By reducing the amount of space lost to small files, the space saving can be even greater.

Another benefit of these compressed files is that disk backup is faster (fewer files in less space). A set of files that could not be copied to a floppy diskette in unpacked form can be joined by ARC and made to fit on the limited disk space.

A trick to adding files to the ARC is to place the files in one subdirectory and use the *a* or *m* switch with a wild-card file name. You have fewer file names to type (*.* versus individual names) and the *m* (move) switch automatically deletes the files after they have been added to the archive.

Another trick is to archive sets of ARC files. Generally, the additional disk saving is less than 10 percent, but combining several archives into one may cut transmission time (when communicating the files by telephone) or reduce "where is it now" file confusion.

Two other major shareware programs are faster and do more compressing than ARC. The programs are half-loaves: one program does the compression-related tasks, and the second program does the extract-related tasks. The first set is ARCA.EXE (add) and ARCE.EXE (extract). The authors are Vern Buerg, a frequent shareware writer, and Wayne Chin.

Another archiver is PKARC.COM by Phil Katz. PKARC.COM builds PKARCA and PKARCE. The ARCA and PKARC programs are comparable, and the shareware community enjoys the friendly competition between the two groups.

These programs are indispensable to those who create many small files or wish to archive a major subsystem (such as 1-2-3 installation programs). The programs should be placed in a \UTILS or \BIN subdirectory.

Improving Disk Performance

As more and more files are created, deleted, and edited on your hard disk, the phenomenon called *fragmentation* begins to steal more of your disk's performance efficiency. No, your hard disk does not disintegrate into tiny pieces, but your files are scattered into many pieces. Your newer files are not stored in contiguous sections of the hard disk. The recording heads must move more frequently to load or store the files. Although the head movements take fractions of a second, the cumulative effect slows the performance of the computer.

Fragmentation is not unusual. DOS uses a first-found, first-used method to store files. As files are deleted and created, gaps occur. DOS fills in the gaps with pieces of files, and your file is strewn across the disk. But your files are complete and called up intact. DOS's built-in sector-accounting system is well-equipped to keep track of the pieces and assemble the file no matter how many pieces the file is in. You lose more performance efficiency, however, over the weeks and months as you use the hard disk. You can even hear the hard disk's recording heads moving back and forth across the disk more times to access a file.

To get a more specific idea of how fragmented your disk files are, run the CHKDSK command (discussed in Chapter 11). NU, the main program of the Norton Utilities, gives you a graphic picture of how each file is stored. Do not be unduly alarmed. If your hard disk is less than 50 percent full or if you've been using the hard disk less than a couple of months, the extent of fragmentation is minimal and you need not be worried—yet. However, if the fragmentation is heavy and interfering with DOS operations, you have two options: Use the DOS commands BACKUP and RESTORE (covered in Chapter 15), or use a utility program to reorganize your hard disk.

Defragmenting with BACKUP and RESTORE

Your first option is the brute-force method. You completely back up your hard disk on a file-by-file basis, reformat the disk, and restore all the files. The reformatting erases all files from the disk and DOS "starts from scratch" as it uses the disk. By immediately restoring your files, DOS stores the files and subdirectories in contiguous sectors on the disk. Your files are stored compactly without gaps, and your disk performance is optimal.

Although this technique lacks elegance, it is effective. This approach has a hitch: the BACKUP-RESTORE process is long. Unless you use one of the hyper-performance programs like FastBack, discussed in Chapter 16 on backup, you may spend two hours or more backing up and restoring files. However, if you want to restructure your disk (move many files or subdirectories) or need to back up anyway, the approach is less than radical.

If you wish to stave off fragmentation longer, a few preparatory steps pay off. When you back up your disk, back up unchanged files first and frequently changed files last. Program files are changed the least; program data files are changed more often. Small data files are next; then spreadsheets, then word-processing documents. Development programs are changed the most frequently. By backing up files in order of the amount of change, you create fewer gaps in the front of the disk. The more frequent gaps are at the end of the disk. Fragmentation, when it occurs, slows your system less because the files are scattered over a smaller portion of the disk.

Also keep together related frequently used files. For example, the 1-2-3 subdirectory with its FONTS subdirectory can be backed up as a group. Even the configuration file (123.CNF) changes infrequently. When you use the program, the recording heads will move less and the files will load faster. However, remember to back up the worksheet and graph directories separately (see the preceding tip).

Another tip is to back up your most frequently used programs first and your least used last. Whenever DOS searches for a program, DOS examines the file allocation table (FAT) at the front of the disk. The closer a file is to the front of the disk, the faster DOS finds that file.

Remember this warning about copy-protected programs installed on the hard disk. You should uninstall all copy-protected programs; back up, format, and restore the hard disk; and then reinstall the programs. This subject is covered in depth in Chapter 15.

Defragmenting with a Utilities Program

The second method involves using one of the public-domain or commercial utility programs that reorganizes your hard disk: CONDENSE from the Mace Utilities, Disk Optimizer from SoftLogic Solutions, and the public-domain Disk OrGanizer (DOG). All three use available RAM and disk space to place your files on the disk contiguously.

The three programs are similar, but their speeds and displays vary. Although all three are considered safe and can handle an abrupt halt (power outage or system reset) without losing files, recovering from a sudden halt is less than graceful. You may need to rename or delete a few temporary files. Although not required, good practice dictates that you back up the hard disk before running these programs.

The Mace Utilities CONDENSE program is a joy to watch. On your screen is displayed a graphical "map" that shows what areas of your hard disk are occupied, already marked off as bad sectors, or open (see fig. 12.3). On a heavily fragmented disk, you see the map covered with "holes," although you cannot yet tell how individual files are split up.

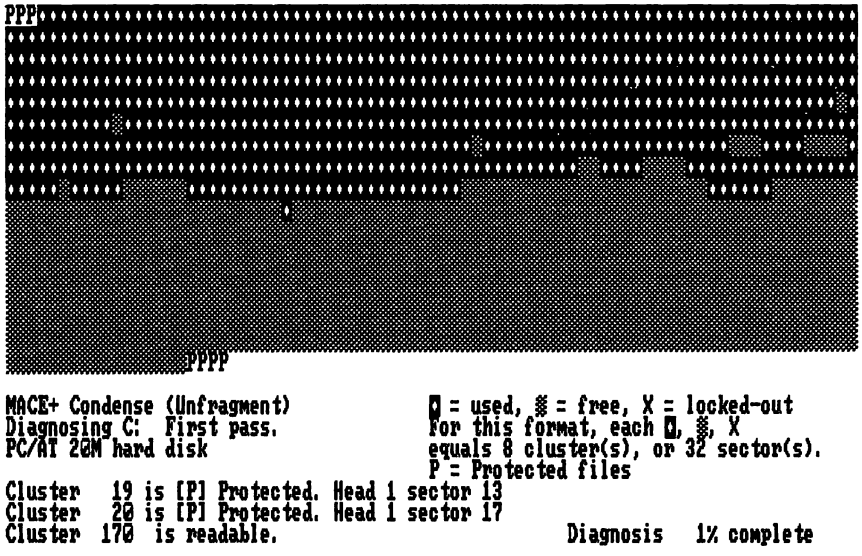


Fig. 12.3. *Mace Utilities screen "map" of a hard disk.*

When the program finishes its diagnostic phase and starts to condense or "unfragment" each file, you see the results on the map. The program plucks each file's scattered pieces and packs them together in contiguous pieces as much in the front end of the disk as possible. On a 20-megabyte disk, depending on how extensive the fragmentation is, the entire process may take from about fifteen minutes up to an hour. Subsequent runs of CONDENSE usually take less time. Unlike the other programs, CONDENSE also repairs damaged files as much as possible.

Disk Optimizer's display is less exciting but just as reassuring. An advantage to this program is the ability to prescan the files. Disk Optimizer gives you statistics on the extent of the fragmentation, optionally on a file-by-file basis. You then decide whether to start or skip the optimization. Disk Optimizer permits you to reorganize files selectively—a feature no other program offers. The average speed of Disk Optimizer compares to CONDENSE.

DOG (Disk OrGanizer) is the slowest of the three. However, DOG has more options for reorganizing the disk. You can tell DOG to reorganize completely, making each file contiguous, or instruct DOG to put the files into pieces as large as possible without moving contiguous files. The additional options allow you to do a quick and fairly complete job or a slow but thorough job.

CONDENSE and Disk Optimizer can be interrupted gracefully. The programs finish the files they are working on at the time and exit, leaving the disk partially condensed.

These programs are best when used about once every month to every quarter. Because these programs take considerable time (fifteen minutes to one and one-half hours), frequent running of the programs is seldom justified. The best time to run the program is when the machine will be unused for a period of time, such as before going to lunch, leaving work, or going to bed.

Safeguarding Copy-Protected Programs

Although copy protection in general seems to be on the wane, some major (and minor) software firms still insist on using it. Most schemes have evolved so that the programs can be installed on the hard disk. However, those schemes use files that you do not see, and the files must be on particular sections of the disk. Disk reorganization may halt the use of copy-protected programs.

The prudent course of action before running any program that plays with your hard disk directory, FAT, or disk space, is first to remove *all* copy-protected programs from your hard disk. All these programs have an "uninstall" program, such as COPYOFF, UNINSTAL, COPYHARD/U. Check the program's documentation for the specifics.

In their documentation, DOG and CONDENSE flatly instruct you to remove copy-protected programs from the hard disk. Disk Optimizer does not issue this caution. Disk Optimizer knows most copy-protection schemes and avoids moving the protected files. However, changes in protection schemes have already bitten SoftLogic's Disk Optimizer once, and a few users temporarily lost

some copy-protected software. SoftLogic immediately corrected the problem, but new schemes may cause similar headaches. If you get a copy-protected program, either check with SoftLogic to make sure Disk Optimizer can be used safely, or uninstall the copy-protected program first.

Using Data PATHs To Trick Programs

Chapter 5 discussed how the DOS PATH command establishes alternative places for DOS to find programs and batch files. PATH has been used throughout the book to conveniently run programs and batch files from any directory.

Recall that PATH applies only to program files and batch files, meaning files with .COM, .EXE, or .BAT extensions. PATH does not solve two critical needs: it will not automatically search auxiliary program and data files and it does not work with supplementary programs that have little or no ability to use the hierarchical directory structure with data files. DOS's PATH command works well with program or batch files and miserably with any other file.

Chapter 9 presented two potential "fixes" for the problem. Both solutions involved using the DOS SUBST command to trick programs into believing that a subdirectory was actually a disk drive. An alternative solution is presented here.

The solution involves using outside programs that trick programs into finding files just as the PATH command makes DOS find files in different directories. In other words, the solution is a PATH-like command for data files. This discussion focuses on two programs, FilePath by SD Associates and DPATH, a public-domain program. Several other commercial, shareware, and public-domain versions of these programs exist, including one sold by IBM's Express Software. Although the syntax and features may vary slightly, all these programs have the same general function.

FilePath

FilePath is the data path program from SDA Associates, the publishers of FilePaq. FilePath is a three-part set. The first part is a device driver, which loads into the computer via CONFIG.SYS. The second part consists of two programs that enable the data path command and set the features to be used. The third part is a configuration file you create with the COPY CON technique, a text

editor, or a word processing program in the programmer's or nondocument mode.

To use FilePath, copy FP.SYS from the FilePath diskette. This file should be copied to the \BIN\DRIVERS or \DOS\DRIVERS directory.

The next step is to place a line in the CONFIG.SYS file to load the FP.SYS device driver. The line should be either

DEVICE=C:\BIN\DRIVERS\FP.SYS

or

DEVICE=C:\DOS\DRIVERS\FP.SYS

depending on where you place the file.

The third step is to establish the FP.CNF file in the root directory of the hard disk. Because the discussion of FP.CNF involves understanding the options to FilePath and its two programs, this discussion will return to the configuration file after covering these two programs.

The syntax of the executable programs is

*dc:pathc***FP** *ED* + - *d1:dpath1/T;d2:dpath2/T;* . . . + - */Dc /Ec /Fc*

d1:dpath1, *d2:dpath2*, and other directories (the . . .) are the paths that hold your data files. FilePath calls this a path list; it represents the directories that FilePath will search. The */T* switch tells FilePath to *traverse* directories, which means FilePath should search the subdirectories of the specified directory. Wild cards are allowed for the directory names.

FilePath has some options that are not preceded by a switch character (/). The three options are:

ED Enter the *edit* mode where a screen editor allows you to change the options. *ED* must be typed in uppercase letters.

\+ Add the specified paths to the list. If the + precedes the path name, the specified paths are added to the beginning of the list. If the + follows the path name, the specified paths are added to the end of the list.

\- Delete the specified paths from the list. If the - precedes the path name, the first matching path name is deleted from the list. If the - follows the path name, the last matching path name is deleted from the list.

Only one of these three options may be given on the command line.

FilePath has other options that influence the DOS options on which FilePath will operate. These options are preceded by a switch character (/):

/Dc Intercept and operate on *directory* searches.

/Ec Intercept and operate on DOS *EXEC* functions.

/Fc Intercept and operate on old DOS *file* opening functions. *c* is either *I*, for intercept (turn on), or *O*, do not intercept (turn *off*).

To trick programs into opening files not in the current directory, the */FI* option causes FilePath to intercept the DOS request to open a file. FilePath checks the list of specified directories and searches for the file. If FilePath finds the file, DOS is tricked into opening the file in the data path directory instead of in the current directory. */FO* turns off the interception of file openings.

FilePath never tricks DOS when a path is given with a file name. If the full file name contains a path character (the \), FilePath passes the unaltered request to DOS.

/DI turns on the tricking of directory searches. DOS has two sets of built-in functions for informing programs what files are in a directory. If enabled, FilePath performs tricks on the old-style DOS directory searches to find files along its data path. FilePath needs this feature to work with programs like MultiMate V3.2 and SmartComm V1.2. */DO* turns the tricking off.

/EI turns on the interception of the DOS *EXEC* function, the DOS facility to load or load and execute another program. Some programs load and execute other .COM and .EXE files. To trick these programs and DOS, you use */EI*. */EO* turns off the "execute" interception.

The following sections discuss the FP.CNF file. This file contains any of the three switches /F, /E, or /D, plus the */L* switch. The phrasing of the */L* switch is

/L=number

where *number* is the number of directory levels that FilePath may search. This information is used by FilePath when the */T* (traverse) switch is given with a directory. You must specify the maximum directory level that FilePath may search. You can find this information by using the TREE or VTREE programs to display the subdirectories and look for the "deepest" directory. *number* is the level number of the directory from the root directory plus one. FilePath calls the root

directory the first level. For the model directory in Appendix A, the *number* to use is 3.

Given the model directory, the FP.CNF file might be

```
/L=3
/FI
/EO
/DI
```

This file specifies three directory levels (*/L=3*), turns on the FilePath trick for file openings and directory searches (*/FI* and */DI*), but turns off the interception for programs that execute other programs (*/EO*).

The FP command that enables FP could be

```
FP C:\WORDS/T;C:\BIN\OVR
```

FP would search the C:\WORDS and all of its subdirectories (the */T* switch) and C:\BIN\OVR for data files.

How would you use a program like FilePath? Use WordStar as an example. To call up WordStar from, and edit documents in, any subdirectory on your hard disk, you would first include in a PATH directory a batch file to start WordStar (say from \WORDS\WS, which holds all WordStar files) like this:

```
C:\WORDS\WS %1
```

This example was used in Chapter 7. To make WordStar find its auxiliary files, the FP command would be

```
FP C:\WORDS\WS
```

To edit a document from C:\ACCOUNTS\NEWYORK, you would perform these commands:

```
CD \ACCOUNTS\NEWYORK
WS
```

That's it. WordStar loads and finds its overlay files (FilePath tricks WordStar into finding its auxiliary files), and you happily edit and save documents. You make no changes with the SUBST command and you don't alter WordStar. FilePath makes WordStar work from any directory.

Data path programs are useful for tricking programs into finding their auxiliary files. For data files you must alter, however, data path programs are less useful. FilePath can intercept and trick DOS into opening and reading a file from any location. But if you must write information to the file, the file is written to the current directory.

For example, you may wonder why I didn't fix the WordStar situation by putting all the subdirectories containing data files in the FilePath command. Setting up FilePath that way gives access to any document, but when you save the document, the file ends up in the current directory. You wind up with two copies of the document, the old one still in its original subdirectory and the updated version sitting in the current WordStar subdirectory. You can use FilePath to trick programs to find their files, but don't use FilePath to trick yourself when you use files that will be altered.

A second executable program is FPSHORT.COM, a shortened version of FP. FPSHORT lacks the editor, the + or – to modify the path list, and the */D*, */E*, or */F* switches. If available memory is tight, FPSHORT can be used to set the path. The directory, execute, and file options must be set with the FP.CNF file. FPSHORT should be used by those short on free memory who do not require resetting the FilePath switches for directory, executable, and file searches.

FilePath is a wonderful program for coping with those programs that are still uninformed about hierarchical directories. Using FilePath helped me get rid of more than 400K of redundant auxiliary files. My favorite technique is to create a directory called \BIN\OVR or \UTILS\OVR. I place the auxiliary files for FilePath's trickery in this single directory. The command to place this directory into the list is

FP C:\BIN\OVR

FP.SYS should be placed in the \BIN\DRIVERS or \DOS\DRIVERS directory. FP.CNF should be created and placed in the root directory of the hard disk. FP.COM and FPSHORT.COM should be placed in \BIN or \UTILS.

DPATH20

DPATH20.COM (short for DataPath V2.0) is the shareware program for tricking programs. Although DPATH does not have the features or functions of FilePath, the program is suitable for users who do not need the extensive features of FilePath.

The program can be executed from the command line or from a batch file. DPATH will automatically accept directories defined by the DOS PATH as the data path, or you can specify a different set of paths. To use a group of directories other than the PATH, use the SET command to place DP in the environment with the appropriate string of paths. For example, the following command sets the

environmental variable DP to C:\WORK1, C:\WORK2, and C:\ACCOUNTS\NEWYORK

SET DP=C:\WORK1;C:\WORK2;C:\ACCOUNTS\NEWYORK

When programs attempt to use a data file, DPATH20 will trick the programs into finding the files in the three named directories.

Like FilePath, DPATH can be turned off at any time. To remove this program, or FilePath, from memory (both programs take about 5K), DOS must be restarted.

DPATH20 does not have the flexibility of FilePath in specifying the path names via the command line or editor. Also, the FilePath programmers have been careful to anticipate technical idiosyncrasies of various programs. If your needs are more demanding or your programs will not work with DPATH, use FilePath.

Providing Security for Your Files

Security takes on different guises in the microcomputer world. The simplest security used to be locks. When hard disks were rare, files stored on floppy diskettes could be simply put away under lock and key. It was safe, easy, and secure.

You can also lock up a hard disk if you have an IBM AT or clone. The so-called "ignition switch" on the left side of the system chassis is one of the simplest ways of securing a hard disk. Turn the switch to the OFF position and, if anyone boots the machine, the power will come on but nothing will work, including the hard disk. An intruder cannot even use the floppy drive to start the machine.

The other way of "locking up" a hard disk is to use a removable cartridge or hard disk configuration. You can lock up a couple of 10 or 20 megabyte cartridges of data as easily as you can floppy disks.

When I talk about security, I mean to differentiate three kinds of violation:

1. Inadvertent or accidental violation
2. Deliberate but mild violation
3. Deliberate and intense violation

In the first case, you want to prevent the misinformed or unsophisticated user from inadvertently deleting or modifying selected files. In the second case, the violator may know a little about computers and may be trying to see or modify privileged information, but is not malicious enough to go to great lengths. The

third type of violation can be any action, including physically damaging a machine, to access or mangle information.

I have no illusions that an AT's keylock could stop a determined lockpicker. Or that five hex bolts on the back panel of your computer will stop someone determined to steal your hard disk. I am talking here more in terms of the first and second kinds of violation.

Other than physically locking up your data or machine, how can you secure your data? The ways include:

- Preventing access to the files or programs in the first place
- Encrypting the files or programs so that access produces apparent garbage
- Leaving behind no scraps that someone could access

Look at each of these.

Prevent Access

One of the first steps you can take is to transfer and lock up the only copy of data. You can copy the more sensitive files onto floppy disks or some other portable medium and lock them away. Copying protects against inadvertent loss. If confidentiality is a factor, you may need to move the files rather than copy them. That is, also delete them from the hard disk once you have verified the accuracy of the copy. That action certainly removes the target of any security problem.

Moving files from the hard disk may be too inconvenient and time-consuming, however. An alternative might be to leave the files where they are and protect them. Every file has associated with it certain file attributes. These file attributes can be altered by a number of programs mentioned in this book (Norton Utilities, PathMinder, FilePaq, PCTools, PCSWEEP, XTREE, and even DOS V3.0 and later).

A file can be changed to read-only, for example, to prevent against inadvertent erasure or modification. Or a file's hidden attribute can be turned on, which prevents the casual user from even seeing the file in a directory. A hidden file can, however, be accessed by EDLIN or the TYPE command, assuming of course that you know the file's name. A hidden program or batch file will also run. (Note: DOS V3.0 cannot modify the hidden attribute yet.)

If you want to prevent misuse of a program, you can put a front-end on it, to step between the user and the operating system. In the

chapter on batch files, a FORMAT.BAT file was described that was created to prevent a hard disk near-disaster. When someone tries to execute the DOS FORMAT command, what they get first is the FORMAT batch file that double-checks the typed command and then passes control to the true format program (which is disguised as XFORMAT.COM).

Batch files are useful for protecting the novices from their own mistakes. The chapter on batch files also discusses examples of menu systems that can be created to guide the novice and control access. You can gain heavy-duty protection by utilizing commercial or shareware menu systems, some of which are discussed in Chapter 13.

Some systems require a password before certain menu choices can be selected or programs run (1 DIR and Fixed Disk Organizer are two examples). Most of these systems can be easily bypassed; but the point is to protect against the casual offender.

Other supposedly more iron-clad systems claim to protect files, programs, and specific storage areas of the hard disk with layers of passwords. I have seen one (Watchdog) in operation. It protects against even the most obvious way of defeating a hard disk security system: booting the system from a floppy disk. I am still a little wary about how it works, especially with copy-protected programs. I am also a little nervous about binding up the hard disk security too tightly.

File Encryption

You can encrypt or encode programs and files instead of, or in addition to, limiting access to them. Then, an unauthorized user can neither interpret nor use the files without specifying a password. Many file encryption products are available (including FilePaq's SDACYPHR, DiskOptimizer, PathMinder, and Superkey).

File encryption is easy to do. And, you can encrypt program files as well as data files. Do not take any chances on copy-protected programs, however. File encryption may very well mess up the copy-protection and make the program unusable.

You do have to remember the password for encrypted files. The password you provide for encryption must be exactly the same (except maybe for lower- vs. uppercase) for decryption. All the products I have seen make it quite clear that there is no way to decode the file without the original password.

You might consider a compromise encryption policy where you copy original files to a removable medium and encrypt the originals

on the hard disk. This gives you the relative convenience of keeping the originals on the hard disk and the relative peace of mind of having readable copies salted away.

Erase the Leftovers

A final aspect of security is dealing with the leftovers. You know that when you erase a file, it is not really erased. You can erase a subdirectory and its files but the data is still there. Writing files subsequently to the hard disk helps get rid of leftovers, but pieces of the original files still float out there. With all the file-unerase and disk-peeking programs out there, even the casual user can poke around in the junk heap and maybe find a little gem.

If you want to make sure that a file or disk is erased, you can use one of the heavy-duty erasing programs like WipeFile or WipeDisk (both by Norton Utilities) or PathMinder's Kill command (which promises to obliterate a file). These programs work just like a simple erase command, but these erase commands are indeed final.

Hiding Programs

At times, you may want to keep certain files in limbo, hidden from the routine operation of DOS. Among programs that are frequently hidden are those that can destroy data, such as FORMAT, SYS, RECOVER, and FDISK. You can hide these programs so that no one can inadvertently invoke them and accidentally destroy files.

An easy approach is to remove the dangerous commands from the hard disk entirely. Some commands that are used infrequently qualify, such as SYS and FDISK. A second option is to hide these files in a directory not on the PATH, such as \DOS\OTHER or \BIN\DISK. You run these programs by giving an explicit path name or by moving into the directory and then invoking the commands. In this way, the programs cannot be run accidentally. Anyone who knows where the programs are kept can still use these commands, however.

Another bit of trickery has already been shown. Simply rename the commands. For example, FORMAT.COM could be renamed XFORMAT.COM. A user would need to know the right name to invoke the program. You also can combine the renaming trick with a driving batch file. An example of using a batch file that checks parameters for FORMAT was presented in Chapter 7.

Unprotecting Copy-Protected Programs

I have made various references throughout this book to copy-protected programs. Few of these references have been joyous, because copy-protection dulls the joy of using a hard disk. Programs that require you to insert a floppy diskette into the drive to start a program make the backup and restore processes clumsy and make the efforts to keep a hard disk organized more difficult.

I understand why software firms clamp copy protection on their creations. The firms are trying to protect their interests from casual copiers and pirates. The legitimate user suffers in the process, however.

Fortunately, many major software producers (with the current exception of Lotus) and many smaller producers have concluded that protection against the dishonest few is not worth inconveniencing the honest majority. Copy-protection is waning as producers begin to believe that major users, like corporations, will respond positively and not stoop to pirating.

Alas, some publishers do not share this view. But, as each new copy protection scheme has been invented, a way of circumventing it appears. Software manufacturers such as Quaid Software (ZeroDisk), TranSec Systems (Unlock), and Central Point Software (CopyII PC) have developed programs that easily puncture many of the copy-protected programs.

These programs either intercept the copy-protection or alter the program to remove the copy protection. The altered copies can be backed up like any normal file.

You should understand, however, that illegally copying software is a crime. Central Point Software warns in bold print that "[making legal backups] is the only right to copy afforded to you under the law." Illegal copying denies the software publisher the proper royalties and eventually removes the impetus to improve programs. Illegal copying hurts the publisher and eventually hurts the legitimate software purchaser.

Despite the legal and technical complexities of copy-protection, programs that get around protection are easy to use. The programs do not work on all software, but when they work, they work well.

An example is COPYII PC, which is as easy to execute as the DOS COPY command. The program copies copy-protected diskettes. To use it, copy COPYII PC to your hard disk (the program is not

copy-protected). Then put the program disk to be copied into drive A. For a single floppy disk drive system, type

COPYIIPC A:

For a system with two floppy disk drives, type

COPYIIPC A: B:

By the way, COPYII PC may not run properly with a patched version of DOS. You should run a vanilla version of DOS without alterations made by RAM disks, print spoolers, or foreign memory boards. Most device drivers loaded through CONFIG.SYS do not interfere, however.

Users with one floppy disk drive are prompted to change from the diskette they are copying to a blank formatted diskette during the copying process. Such users should write-protect the original diskette before beginning to copy. Users with two floppy disk drives are instructed to place the diskette to be copied in drive A and the blank formatted diskette in drive B. COPYII PC makes a carbon copy of the original diskette in a minute or two.

During the copy process, COPYII PC may report on "verify errors," probably used by the copy-protection. (Many copy protection schemes fiddle with the logical organization of a disk.) The program asks you if you wish it to correct these errors. Usually, the answer should be *No*. You want to duplicate exactly the original disk's peculiarities.

After COPYII PC has finished, exit the program, put the original diskette away, and try the duplicate. If it works, you have backed up your diskette. This solves the problem of making a backup, but it does not solve the problem of running a key-disk program without inserting the floppy diskette.

NOKEY is a companion program that accompanies COPYII PC. NOKEY intercepts the call by the copy-protected program to check the key diskette. You first run NOKEY; then you run the copy-protected program without the key diskette in the disk drive. If successful, NOKEY can be used before you run the program and the key diskette is not necessary. NOKEY does not work with all copy-protection schemes, however, and some programs (such as Sidekick or DOS's Verify) clobber NOKEY. You are then required to rerun NOKEY before you use the copy-protected program again.

Newer copy-protection schemes allow the program to be installed on the hard disk and run without a key diskette. These are the programs that must be uninstalled before you back up the entire

hard disk. You may not be able to use or uninstall the program if you forget this step.

Another COPYII PC program, NOGUARD, which was originated by a company called Softguard, also can be used with this type of copy-protection. NOGUARD can turn some installable programs into "normal" (non-copy protected) versions that can be run and backed up. Like NOKEY, NOGUARD works on many but not all programs.

The instruction manual for COPYII PC is mercifully brief and to the point. Central Point Software also has a policy of selling upgrades at any time for a small fee. As copy-protection schemes change dynamically, the policy is beneficial for those needing that latest version of the program.

Although CopyII PC does an excellent job, Central Point also offers the COPYII PC Option Board for particularly nasty copy protection schemes. The board plugs into any slot, and you run the cables from the disk controller board to the Option Board. You use the provided cable to connect the Option Board to the disk drive. The Option Board is a different type of disk controller and can copy virtually any software. The only scheme the board cannot defeat is Vault Corporation's ProLock scheme, where a laser burns a microscopic portion of the diskette. The Option Board cannot duplicate that physical damage.

Other products that handle the copy protection problem include TranSec Unlock (which focuses on a handful of popular programs and guarantees to unlock only those), CopyWrite (another copy program), and Quaid Software's ZeroDisk, which works like NOKEY.

The controversy surrounding the copy-protection issue continues. Lotus and Ashton-Tate have challenged a few major corporations for using illicit copies. As of this writing, Quaid, a Canadian company, is being sued by Vault under Louisiana's "shrink-wrap" law for allegedly disassembling its copy-protection code.

The caution should be: unprotect only software you have purchased. Don't injure the software industry and other users by making and using illegal copies.

Using Large Disk Drives

Chapter 2 on DOS and Chapter 3 on selecting a hard disk alluded to the fact that the ROM BIOS and DOS do not natively recognize all disk drives. If the computer does not "know" the hard disk, the hard

disk cannot be used correctly. A second problem is that DOS cannot natively address disk drives larger than 32 megabytes.

Disk drives have different characteristics. The number of platters and recording heads can vary. The number of cylinders used can vary. Usage information, such as how long the wait should be before sending information to the disk drive or until the recording heads have settled on the proper cylinder, varies between disk drives. Disk drives with the same storage capacities are not alike. For example, a 20M disk drive can have four heads (two platters) with each platter holding 10M or can have eight heads (four platters) with each platter holding 5M.

To use a disk drive correctly, the computer must know exactly how many cylinders and heads the disk drive has and other usage information. A table for known disk drives is maintained in the ROM BIOS. The XT class of computers knows some 10M, 15M, and 20M disk drives. The AT class of computers knows about 30 different disk drives. If the ROM BIOS of the computer does not know the disk drive, the computer needs outside help to use the "foreign" disk drive.

DOS V3 and earlier has a magic limit of 32M. Microsoft made one critical assumption when initially writing DOS. The assumption pervades all parts of DOS. The assumption was that a disk has no more than 65,536 sectors. As most hard disks use 512 bytes per sector, the maximum size is a limit of 32 megabytes.

Given that people do use disk drives larger than 32M, what is the magic used to break this ROM BIOS and DOS barrier?

The magic has several potential solutions. The most popular solution involves tricking DOS into seeing the very large hard disk as several separate disk drives. Another solution is to modify DOS itself so that larger disk drives can be used.

Subdividing one physical disk drive into several logical (apparent) disk drives is an established procedure. FDISK performs the task of dividing a disk drive into partitions. We normally use the partitions so that each different operating system "sees" only its partition. The other partitions owned by other operating systems are invisible.

Rather than dividing the partitions among operating systems, a single operating system can be tricked into seeing each partition as a separate disk drive. Large disk drive partitions can appear to DOS as separate physical disk drives. The heart of the magic is the right device driver, which is installed via the CONFIG.SYS file. The right device driver also overcomes any recognition problems posed by the computer's ROM BIOS.

A second method, often used in conjunction with the previous method, is to change the size of a sector for the hard disk. By increasing the size of a sector, disk drives as large as 130M can be used. The change is made by directly modifying the BIOS of DOS or by using a device driver.

Two packages that help in using very large disk drives are Vfeature Deluxe from Golden Bow Systems and SpeedStor from Storage Dimensions. Both packages have the necessary utilities to rearrange DOS's attitude about using many megabyte disk drives.

Vfeature uses both the partitioning and increased sector size approaches. To use very large disks as a single disk drive, the size of the sector is increased. Sectors can be up to 16,384 bytes each, allowing 1 gigabyte (one thousand million byte) disk drives.

Vfeature also offers programs to partition the hard disk into multiple volumes which appear as separate disk drives to DOS. Vfeature has an option called spanning, the exact opposite of splitting a single disk into multiple volumes. Spanning combines two separate disk drives so the two appear as one unit to DOS. For example, two 140 megabyte disk drives can appear as a single 280M disk drive to DOS. However, you cannot choose both types of operations (or need to, for that matter).

Vfeature programs can logically format each volume (DOS's FORMAT cannot cope with large disk drives). Vfeature incorporates various security options for each volume. Volumes can be selectively enabled (mounted) or hidden from normal operation (unmounted). Each volume can be write-protected. Each volume can be password-protected.

SpeedStor is a combination disk test and multiple volume maker. SpeedStor incorporates many of the function of Kolod's HTEST/HFORMAT. Hence, SpeedStor can be used to physically test and format any hard disk drive and compares favorably to HTEST/HFORMAT.

SpeedStor, like Vfeature, uses the partition and increased sector size approach. SpeedStor has an FDISK-like function to create eight different partitions on a hard disk. Each DOS-used partition can be bootable, compatible, or a DOS extension partition. If DOS is to be started from the hard disk, the first partition must be the bootable-type and cannot exceed the 32M limit. DOS compatible partitions also are restricted to the 32M limit, but can be any partition, not just the first partition. The extension partition can be any size and is not affected by the 32M limit. SpeedStor exceeds the single disk drive

limitation by using sectors of up to 16,384 bytes each, also allowing 1G disk drives.

Both programs include a device driver which must be included via the CONFIG.SYS file. When DOS boots, the FIXT_DRV.SYS file (for Vfeature) or the HARDRIVE.SYS (for SpeedStor) is read into memory and installs itself into DOS. The device driver does the necessary work to translate what DOS thinks it sees (large sectors or multiple disk drives) into what the computer really uses.

The two packages are comparable. If you like the added features of joining two disk drives into one or the additional security features, Vfeature is the choice. If you'd like an extensive disk test and physical hard disk format program, SpeedStor is the appropriate choice.

Golden Bow, the authors of Vfeature Deluxe, also publish several other packages. Vcache and Vket are hard disk and floppy disk caching programs, respectively, comparable to Lightning and Flash (mentioned earlier in this chapter). Other products include Vopt, a fast disk-optimizer, and Vscreen, a speed-up program for video displays.

Since both products alter DOS's view of the hard disk, the products are best kept on diskette. The exception is the device driver for each product, which should be placed in \BIN\DRIVERS or \DOS\DRIVERS.

Using DOS's International Functions

In Chapter 4, the discussion about establishing the CONFIG.SYS file referred to the international aspects of DOS. The country code was incorporated in MS-DOS V2.11. The keyboard and graphics routines were previously provided only to international customers. Starting with DOS V3.0, all DOS purchasers receive these routines.

The first international programs are a set of keyboard programs. These programs supplement the ROM BIOS of the computer to accept and display international characters. The basic set of programs (although many manufacturers add additional programs) are

Program	*Country*
KEYBFR.COM	France
KEYBGR.COM	Germany
KEYBIT.COM	Italy
KEYBSP.COM	Spain
KEYBUK.COM	United Kingdom
KEYBUS.COM	United States

The easy way to run these programs is to place a line in the AUTOEXEC.BAT file to invoke the proper keyboard program. If these programs have been placed in the \BIN or \DOS directory, add the line with the appropriate command *after* the PATH has been set. If the PATH command establishes the \BIN or \DOS directory, you need not give an absolute path name with the command. For example, you can execute the Italian Keyboard program by simply using the line

KEYIT

instead of

C:\DOS\KEYIT

If you are using the U.S. keyboard, you need not use the KEYBUS program. In fact, the program is a waster of memory because the keyboard loads into memory and stays, stealing about 2K of RAM.

These memory resident program may conflict with other keyboard enhancers, so test your favorite keyboard macro-maker (such as ProKey, Superkey, or CED) before you make this program a regular procedure in your AUTOEXEC.BAT file.

You may encounter a small problem with the IBM Color Graphics Adapter (CGA) and its clones when you use foreign character sets.

The graphics characters (which include the foreign language characters) are in the upper-ASCII set (ASCII values greater than 128). The CGA smears these characters, making them almost unreadable. The problem occurs only when you use the medium-resolution, 320-by-200, 4-color mode; it does not occur in normal alphanumeric text mode or in the high-resolution, 640-by-200, 2-color mode.

IBM provides a fix for this problem with a program called GRAFTABL.COM, which is on the DOS system master diskette. When loaded, the program establishes an alternate character set in RAM and then directs the CGA to use this set. The medium-resolution characters become legible.

To execute GRAFTABL, you type

GRAFTABL

This program, like the KEYBxx programs, should be run only once after startup because each invocation steals about 1200 bytes of RAM. Additional copies of GRAFTABL are a waste of RAM space; no additional functions are added.

For conservative RAM users (those who do not wish to waste any RAM space), GRAFTABL should be run only if you have programs that use ASCII characters with values greater than 128 in the medium-resolution graphics mode of the CGA. If you run programs like this often, place the GRAFTABL statement in your AUTOEXEC.BAT file after the PATH line.

Updating to a New Version of DOS

As new versions of DOS are developed, you may choose to upgrade your computer. The current version of DOS is V3.2. Microsoft has licensed some European companies to publish MS-DOS V4.0 for their machines, and the IBM/Microsoft team are working on DOS V5.0 and V6.0.

To update your hard disk to a new version, you generally do *not* need to repartition or reformat the hard disk. Nor do you need to copy all your files on the disk, upgrade, and then reload the files. In most cases, you can upgrade DOS without disturbing your other files. Good practice still dictates backing up the hard disk *before* upgrading, however. Paranoia can be healthy.

On occasion you must reformat the hard disk. The indicator comes when you place the operating system on the hard disk. That problem will be discussed in detail at the appropriate step.

To upgrade, you will need to boot the new DOS, test your established programs, place the DOS system files on the hard disk, and then replace the old DOS utilities files with the new ones. The final step will vary between the DOSs.

Step 1. Boot the computer using the new version's DOS Master diskette.

Make a copy of the new master diskette immediately. To boot, put the new DOS master diskette into drive A and start your system.

Using DISKCOPY, make a copy of the diskette. If your computer has one floppy drive, type

DISKCOPY A: A:

If your computer has two identical floppy drives, type

DISKCOPY A: B:

Follow the directions on the screen. Remember that the source diskette is the DOS System Master. The target diskette is the blank diskette that will hold the copy.

Step 2. Test your applications to ensure that they work with the new DOS.

Before you place the new DOS on the hard disk, test your major programs. The testing should take from one to three hours. The programs should work, but test them anyway. Your tolerance for surprises may be higher now than it will be later.

If your programs depend on certain CONFIG.SYS or AUTOEXEC.BAT settings, copy the two files from your hard disk to the working master diskette and edit the files as needed. You may need to add a new CONFIG.SYS command or change references from drive C to drive A. Reboot the system from the working diskette and then exercise your programs. If problems arise, contact the publisher before placing the new DOS on your hard disk. If no problems show, proceed to the next step.

While testing your programs, do not use the older DOS housekeeping programs such as CHKDSK or FORMAT. The problems that you can encounter using an older DOS's utilities with a newer version of DOS range from no problem at all to total file mayhem. Use only the new DOS programs from the diskette.

Step 3. Place the new copy of the operating system on the hard disk.

The SYS command places the IBMBIO.COM and IBMDOS.COM (or MIO.SYS and MSDOS.SYS) files on the hard disk. With the working copy of your DOS master diskette in drive A, type

A:SYS C:

This command instructs DOS to add or replace the system files (the hidden ones) on your hard disk with the ones from the startup diskette. When you see the message `System transferred`, the crucial

step has been accomplished. You may continue with the remaining steps.

You may, however, see one of these messages:

```
No room for system on destination disk
Incompatible system size
```

If so, stop! The new system files are larger than your older DOS system files. The hard disk does not have enough reserved space for the new DOS system files. You *must* back up your hard disk, logically reformat the hard disk (but you need not run FDISK again), and restore the files. Then you can go to step 5. When you format the hard disk, give the */S* (system) switch. This will automatically copy COMMAND.COM to the hard disk.

Step 4. Copy the new COMMAND.COM to the hard disk.

SYS does not automatically copy the third important DOS system file; you must do this manually.

Step 5. Replace all previous DOS files with the new versions.

For DOS versions earlier than V3.2, use the COPY command and diligence to make the replacements. For DOS V3.2., use the REPLACE command.

For pre-V3.2 DOS, copy the files from the DOS Master diskette into the directory that holds the old DOS files. If you have followed the techniques described in this book, the DOS files should be in the BIN, DOS, and/or UTILS directories. If you have renamed commands, delete the old copy and rename the new file.

Retain your old CONFIG.SYS and AUTOEXEC.BAT files. Do not copy these files from your working copying of DOS. The disk drive and path names would need to be changed again.

If you place your device driver files in a separate directory, such as \BIN\DRIVERS or \DOS\DRIVERS, move the appropriate files from BIN or DOS to DRIVERS.

Remember also to copy the files from the supplemental diskette(s) to your hard disk. Most .COM and .EXE files will go to the BIN or DOS subdirectory. BASIC sample files would go to \BASIC\SAMPLES or whatever directory holds your BASIC programs. If you see .PIF files, place these in the directory that holds files with the same root name. For example. BASICA.PIF goes wherever you place BASICA.COM.

For DOS V3.2 and later versions, you can take advantage of the REPLACE command, which selectively replaces files anywhere on the hard disk. In other words, REPLACE can traverse the directory structure to replace files.

With the copy of the original DOS master diskette in drive A, type

REPLACE A:*.* C:\BIN /S

or

REPLACE A:*.* C:\DOS /S

The commands operate on the files in the \BIN or \DOS branch of the directory tree. These commands replace any existing DOS file with the file from drive A that has a matching name. The /S switch tells REPLACE to search all subdirectories in the given path.

The compelling reason for placing all DOS files into one branch of the subdirectory system is the impact of the REPLACE command. All DOS files are in the BIN or DOS subdirectory, or subdirectories of these directories. The REPLACE command hunts down and replaces all old copies of DOS files with the new copies.

There is a single caution in using REPLACE. REPLACE compares file names, not contents or dates. The first time I used REPLACE, I used the command

REPLACE A:*.* C:\ /S

In an off-the-beaten-path directory sat a program called SORT.EXE, a version of SORT that was superior to the DOS SORT. REPLACE ambled down into this subdirectory, saw a matching name to SORT.EXE on the DOS master diskette, and replaced my private SORT.EXE with another copy of DOS's SORT.EXE.

So, never unleash REPLACE without making sure that it will replace only the needed files.

Having replaced the proper files, you add new files to the disk by typing

REPLACE A:*.* C:\BIN /A

The */A* switch adds files that do not appear in the destination directory. REPLACE copies to the destination directory the files whose names are unique (that is, that do not appear in the destination directory).

Now insert the supplemental diskette(s) and repeat the basic process. This time give specific names because you want the files in

different directories. Given that the sample BASIC programs go into \BASIC\SAMPLES, you would use these commands:

```
REPLACE A:\*.COM C:\BIN /S
REPLACE A:\*.EXE C:\BIN /S
REPLACE A:\*.PIF C:\BIN /A
REPLACE A:\*.BAS C:\BASIC\SAMPLES /S
```

These commands copy the new files into their correct directories.

Summary

This chapter presented programs and techniques for making your computer system faster or easier to use. The high performance items are RAM disks and disk cachers. If only one of the two can be used, choose a disk cacher. Better performance can be obtained by using both RAM disks and disk cachers.

The chapter discussed hiding a program and using a data path with auxiliary files. Keyboard enhancers and graphic programs for international computers were described. This section also explained how files take more space than reported by DOS and how archivers can pool many small files into a large file that conserves disk space. The discussion also included ways that disk optimizers can defragment files and improve your disk's performance.

Part 3 of this book concludes with this chapter. Part 4 begins with a look at hard disk managers and windowing environments. You will explore the vital issues of backing up the hard disk and optimal ways to set up many popular programs on a hard disk

Part IV

Improved Hard Disk Management

Part IV builds on the techniques and ideas presented in the first three parts of the book and presents more programs, techniques, and ideas for enhancing your use of the hard disk.

Chapters 13 and 14 cover hard disk managers and environments. These programs offer friendlier or more efficient ways to manage your disk or to run programs on the hard disk.

Chapters 15 and 16 deal with backing up the hard disk. Chapter 15 explains the basics about backup and covers BACKUP, RESTORE, and XCOPY—the three DOS programs that handle backup. Chapter 16 presents programs and additional hardware that make the backup chore easier or faster, or both.

Chapter 17 gives information for installing a number of popular programs. The material in this chapter reflects the philosophy and suggestions presented throughout the book.

You are encouraged to examine carefully Chapters 13 through 16. The most critical chapters are those on the subject of backup, the task that left undone can produce dire consequences. You probably will want to skim Chapter 17 on program installation. Although all popular programs cannot be covered in this book, you may find the techniques discussed for similar programs applicable to the programs you use.

Chapter 13

The Hard Disk Managers

As described in Chapters 7 and 8, batch files, either provided with the applications programs or created by yourself, can make your work easier. You can use batch files to navigate around your hard disk, delete and copy files, run programs, and generally manage your hard disk. But batch files are just a start. The arrival of hard disks on the scene brought a slew of programs created to make managing a hard disk even easier. And the flow of programs continues.

With hard disks getting cheaper and more popular, more and more people are being confronted by hard disks with—or instead of—floppy disk drives. Recognizing this development, software firms are producing utilities to help both neophytes and advanced users. For the neophyte, utilities are perhaps a way to avoid the sometimes cryptic, sometimes archaic, language of DOS. For the advanced user, utilities facilitate routine tasks that need to be done but take a great deal of time.

The broad category of hard disk management software includes menu-development programs, file managers, and even program environments.

The programs with menu-development tools help you create hierarchical menus. These menus guide users through the various applications and DOS commands; the users choose options and implement commands with only one or two keystrokes.

File managers are programs that deal primarily with file management—copying, moving, weeding out, creating passwords, editing, printing, and generally straightening up the masses of hard disk files in many layers of subdirectories.

Program environments combine file management tools with advanced capabilities like windows; macros; on-line desktop utilities; multi-tasking, or concurrent operation, of programs; and data transfer between applications. Products of the third type are reviewed in the following chapter.

This chapter gives you a look at some of the popular hard disk management software packages. Obviously, one chapter of this book cannot cover every available program. The products reviewed in this chapter are the Fixed Disk Organizer, 1 DIR™, XTREE™, PCSWEEP™, The Norton Commander™, PathMinder™, and PCTools™. Also discussed are the public-domain programs AUTOMENU and G-WHIZ. Besides giving you an idea of what these programs do, the explanations alert you to some of the characteristics you should look for when considering the purchase of these and other types of programs.

Fixed Disk Organizer

The Fixed Disk Organizer (FDO), which came out in 1983, represents an early model of this type of software. This granddaddy in the hard disk management area is a program produced by one of the pioneers in the computer field—IBM itself.

FDO is a good organizational tool. It provides easy setup of hierarchical menus that can call up and execute programs or combinations of DOS commands. Figure 13.1 shows a typical menu screen of FDO. You can structure your hierarchical menus with separate screens, or "pages," of related menu items grouped together and with menus leading to other menus.

The FDO structure also automatically provides for the inclusion of on-line help screens. With your own word processor or text editor, you create the text for a help screen or other screens relating to any option on a menu or submenu. The menu-editing function of FDO lets you attach this text file to the appropriate menu option.

Each option in a menu can lead to either another menu or a batchlike series of commands that you write. You type these commands within the simple FDO editor. FDO then automatically saves the commands and the menu hierarchy in its own special file.

```
                                Master Menu
Date: 08-17-1986                                          Page 1 of 1
Time: 16:38:31                                            Version 1.00

          1. Word Processing
          2. Electronic Spread Sheets
          3. Communications
          4. Business
          5. Development Tools
          6. DOS Commands

 ↑ or ↓ point to option     F1 Help
                            F7 Quit
 ENTER select option        F9 Menu Maintenance
```

Fig. 13.1. *Typical FDO menu screen.*

You can specify a password if you wish to limit access to a particular menu option. This password also is needed to edit the menu item.

FDO fills a gap in DOS's batch file system with an advanced feature of its own—the use of variables in DOS commands. From reading the discussion of batch files in Chapters 7 and 8, you recall that the DOS batch file system does not support interaction with the user much beyond "Strike a key when ready." With FDO, you can have much more specific prompts; in fact, you can have any prompt you wish. For example, you can ask the user to

```
Indicate which drive your data files are on...
```

or

```
What name do you want to use for the output file?
```

The responses can be saved and used in another place in the series of DOS commands initiating, executing, or finishing an application. This capability makes FDO a good tool for creating interactive menus or other batch files.

The program provided by IBM to install FDO on your hard disk is simple to use. Basically, FDO copies many files from the FDO floppy diskette to your hard disk. These files include the menu execution program itself and a sample start-up set of menu screens, which—of course—can be modified later. Actually, you may want to erase many of the menu files because you may not need them for your work.

The important point about FDO installation is to make sure that you first create a separate subdirectory (say you name it \FDO). Next, make that subdirectory the current one by typing

CD \FDO

at the C> prompt. Then run the installation program. It will place all the FDO programs into the \FDO subdirectory. This method makes sense in most cases anyway. Using a separate subdirectory keeps your root directory clear and is a good organizational tactic.

You have another reason for creating a separate FDO subdirectory and making it the current directory. As part of the installation process, FDO places in the default directory its own AUTOEXEC.BAT file, which starts FDO. If you install FDO in your root directory, FDO replaces your present AUTOEXEC.BAT file with its own. (Recall that the AUTOEXEC.BAT file is the one that DOS looks for and automatically executes whenever you turn on the machine or reset with Ctrl-Alt-Del.)

You may have in your AUTOEXEC.BAT file some handy commands that do special jobs for you—perhaps set an internal clock, set up a PATH, or change the prompt characters. Because FDO insists on replacing your AUTOEXEC.BAT file, those handy commands are wiped out. Of course, you can go back and modify the AUTOEXEC.BAT file after FDO is installed. Every time you use the FDO program, however, your AUTOEXEC.BAT file is replaced by the FDO version.

If you place FDO in its own separate subdirectory, FDO cannot interfere with the AUTOEXEC.BAT file in the root directory of your hard disk. If you also want FDO to spring into action whenever you boot your system, add the following lines to the end of your own AUTOEXEC.BAT file:

CD \FDO
AUTOEXEC

Your AUTOEXEC.BAT file will then start FDO's start-up program.

FDO doesn't help with any of the actual hard disk file management tasks. This crisply working utility is strictly a menu program to help you run programs and execute DOS commands.

1 DIR

1 DIR from Bourbaki, Inc., is perhaps one of the oldest non-IBM entries in the hard disk management field. By creating a *shell* to

cushion you from the rigors of DOS, 1 DIR helps in several areas: disk file management, batch file creation, and menu creation.

The shell fits "around" DOS, in the sense that its display leaves room at the bottom of the screen for the DOS prompt (see fig. 13.2). You can take advantage of 1 DIR's features and still have access to the usual (familiar?) DOS level to do things that you cannot do in 1 DIR. When running, 1 DIR steals only about 50K from your machine's available RAM.

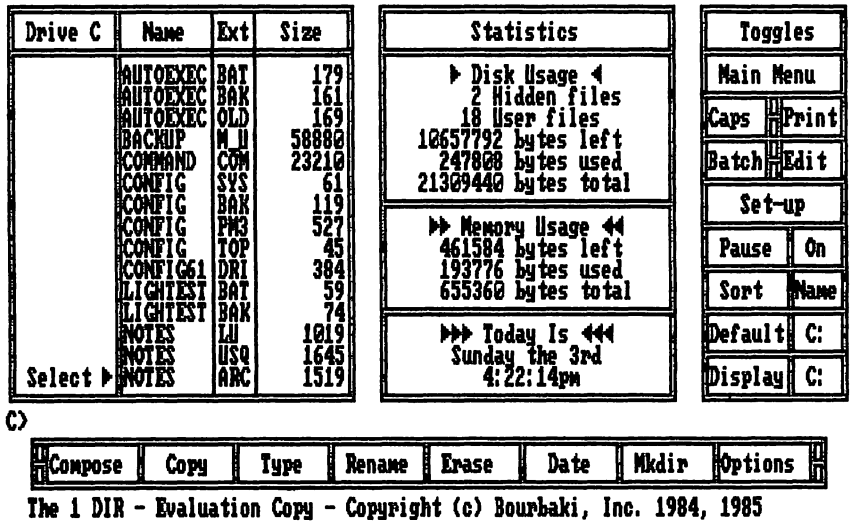


Fig. 13.2. *The display of 1 DIR.*

1 DIR displays a statistics window (which shows the current disk usage, memory usage, date, and time); a files directory window (which shows the files in the current directory and a file cursor used to operate on a file or subdirectory); a command menu along the bottom of the screen; a setup menu (to modify the display); and a toggles section (which indicates the on/off status of various conditions). Between the files directory window and the command menu is the DOS prompt, awaiting your commands.

You can move up or down the subdirectory tree just by pointing to the appropriate subdirectory in the files directory window. You can also "compose" DOS commands by moving the cursor through the command menu and pointing to the files to be operated on. A press of the plus (+) key on the numeric keypad brings the selected file name down to the command line; pressing the minus (–) key brings down the file extension, if needed. For massive file operations

(like a COPY) in which you want to specify a large group of files, you easily can tag a group of files to work on.

To substitute for or complement the 1 DIR menus, you can create and put in place alternate menus. (The technique could be used to add some DOS commands not already included, like the often neglected but useful COMP command.) The menu builder takes a little while to get accustomed to, but it can be used to create a complex series of menus and commands.

The menu hierarchy has password security provisions so that users can be restricted to certain menus. Certainly, this system is not heavy duty security to keep out the intentional troublemaker. The system is meant only to keep well-meaning users out of trouble.

1 DIR also provides a batch file creation mode, which assists you in building up a batch file from scratch. The program includes a capture mode that lets you create a batch file as you execute the instructions within it.

One of the little bonuses to this program appears (disappears, actually) when you don't touch the keyboard for a few minutes. The program automatically blanks out the screen (which may cause you panic the first time). This feature prevents the "burn in" of characters whose image stays on the screen too long. The experts warn against burn in, which might result from leaving the machine on but unattended for a while. This recommendation does not apply to color monitors. They use a different type of phosphor, which does not burn. Of course, you can always turn down your monitor's brightness control when you leave the computer for a while, but you have to remember to turn the control.

You may have 1 DIR install itself or you can override the process for your special needs. By itself, 1 DIR creates a 1 DIR subdirectory and copies the menu and option files; then 1 DIR copies the main operating programs to your root directory.

You can place the 1 DIR subdirectory anywhere in your subdirectory hierarchy (other than emanating from the root). The manual explains how to use the DOS SET command to tell the 1 DIR program where the subdirectory is located so that DOS can find the files it needs. If you like keeping your root directory fairly clean, you can move the program files from the root directory to the 1 DIR subdirectory (called 1 DIRDATA) and the batch files that start up the parts of the program to a separate \BATCH subdirectory. Then tack the 1 DIRDATA directory onto the end of your PATH command.

This discussion has only touched on the program's capabilities. 1 DIR is a well laid out package, although you do need a bit of time

to get used to all the special keys. For the most part, they are mnemonic (Ctrl-F to Flag files, for example), but remembering them all takes a while.

As mentioned, the comfortable part of the program is that it is truly a shell that fits around DOS but doesn't obscure the DOS prompt itself. If, for old times' sake, you want to execute a DOS command or an applications program directly, you can just type the command as if 1 DIR weren't there at all. 1 DIR disappears, your application pops up as usual, and you do your work. Then when you exit from the program, you are thrown back to—the DOS prompt? Oops, the DOS prompt is replaced by the phrase `Press a key to continue...` Pressing a key brings back the 1 DIR screen. The shell is there to assist you but not get in your way. Pressing Ctrl-C removes the shell from memory entirely.

1 DIR is a useful program because it simplifies disk operations but does not hinder your access to DOS itself.

XTREE

The program XTREE takes an approach slightly different from 1 DIR and FDO. XTREE is nothing more—but also nothing less—than a zippy hard disk file manager. XTREE does not come with a menu creation module or a built-in text editor to help you create batch files. XTREE, which is produced by Executive Systems, Inc., does a straightforward focused job extremely well.

What is this job? XTREE helps you manage the subdirectories and files on your hard disk. From the moment you boot up, you get a vivid bird's eye view of both the overall hard disk structure and any particular subdirectory you want (see fig. 13.3). As you move the cursor through the top (overall) view, the bottom view changes to show the individual files included in the highlighted subdirectory. The right side of the screen displays some total-disk and specific directory statistics.

You can easily jump between the top view and the bottom view if you need to operate on an individual file rather than an entire subdirectory. The command menu changes accordingly to show the commands applicable in the individual file mode.

XTREE also provides various ways to tag files for mass operations like COPY, DELETE, MOVE, and RENAME. In addition to stepping through and tagging individual files, you can tag all files that satisfy a wild-card designation (*.WKS) in one or all subdirectories. You also

```
Path: \

\
├─123
├─BATCH
├─BOOK
├─CMASTER
├─DOCS
├─DOS31
│  └─BASICPRO
├─DU
├─FASTBACK
├─INSET
├─IOMEGA
├─IRWIN
└─JUNK

ARC2    .99        BACKUP  .M_U       CONFIG  .SYS
AUTOEXEC.BAK       COMMAND .COM       CONFIG  .TOP
AUTOEXEC.BAT       CONFIG  .BAK       CONFIG61.DRI
AUTOEXEC.OLD       CONFIG  .PM3       IBMBIO  .COM

FILE: *.*

DISK: C: **DOS 3.1**
 Available
   Bytes:10,668,032

DISK Statistics
 Total
   Files:        669
   Bytes: 9,786,399
 Matching
   Files:        669
   Bytes: 9,786,399
 Tagged
   Files:          0
   Bytes:          0
 Current Directory
   \
   Bytes:    223,391

DIR      Available  Delete  Filespec  Log disk  Makedir  Print  Rename
COMMANDS  ^Showall  ^Tag  ^Untag  Volume  eXecute
 ↑↓ scroll  RETURN file commands   ALT menu          F1 quit F2 help
```

Fig. 13.3. *XTREE display.*

can tag, in a subdirectory or the entire disk, files that either have or don't have a particular file attribute (hidden, system, read-only, or archive). This powerful combination of features gives you greater flexibility in file operations.

As you tag individual files or groups of files, the stats window on the right side of the screen updates to show you how many files and how many bytes are involved for the selected group. The program also provides additional functions that improve on DOS, like a simple VIEW function so you can page back and forth through a text file rather than use the DOS TYPE command.

The documentation provides a straightforward example of using the tagging feature to smooth the process of making backups. Although not intended primarily as a backup program, XTREE's tagging and COPY functions can act as a reasonable substitute for the DOS COPY command. For example, to make an incremental backup, you use one command to tag the files that have been created or modified since the last backup set was made. A second command copies all the tagged files in one operation. A third command then marks those files as backed up so that they are not backed up the next time you execute this operation (unless, of course, the files were modified in the meantime).

For this operation, you would probably use the second of the XTREE copy commands. This command duplicates the subdirectory

structure on the disk you are copying to. If a file being copied is in a subdirectory called \DOS\BASIC, the command creates that subdirectory on the target disk (if the subdirectory doesn't already exist) and copies the file into the correct subdirectory. This feature helps you know where to restore the file, if necessary. Note that neither of XTREE's copy commands splits files between disks. Therefore, files larger than the diskettes you are copying to cannot be copied.

If you want to run a program, you can jump out to DOS at any time and still keep XTREE resident in the background. You know the program is there because the DOS prompt and cursor are a little different. The prompt character is now > and the cursor is blinking. You also note that some of your computer's memory is used by XTREE. You can run any programs you want and then easily return to XTREE when you are through.

You actually have two ways of temporarily leaving XTREE. The first way leaves XTREE occupying about 50K of RAM and requires XTREE to rescan your hard disk when you return to the program from DOS. The purpose of this rescan, which can take 15-25 seconds, is to reestablish which files are in which subdirectories and so on (just in case you changed some things while in your applications program). When you use the other temporary exit method, XTREE does not rescan the disk. However (contrary to the documentation), this alternative requires more (not less) memory—about 60K—and can invite errors or at least temporary confusion if you modify or move any files on your hard disk while you are working in DOS.

Installing XTREE is a snap. You just copy its few files to an XTREE subdirectory on the hard disk and, if you have a standard machine configuration, type *XTREE* to start the program. If need be, you can use the installation program to customize the program to your particular needs.

XTREE's error trapping seems quite good. For instance, XTREE does not allow you to type a slash character (/) in a file name. If you type an illegal character, the system doesn't beep. Instead, XTREE just waits for you to regain your composure and press the correct key or call for help through XTREE's built-in help screen.

One small complaint is that XTREE doesn't do anything to satisfy the need for a COMP command. You need to jump to DOS to execute COMP. But, all in all, XTREE does a slick job of file and subdirectory manipulation.

PCSWEEP

PCSWEEP is what you might call a quick-and-dirty hard disk file manager. This program from Koch Software Industries does the job adequately with no extra frills and no fancy screen display. PCSWEEP apparently has its roots in the world of CP/M (an operating system predating PC DOS and the IBM PC).

Unfortunately, PCSWEEP's dated roots show in the archaic screen display. After you start the program, you see a menu screen that displays all the commands, which are executed by function-key or Alt-key combinations (see fig. 13.4). Fine, no problems there. Once you press a key to continue, the program jumps to its main display, the beginning of a list of the files in the current directory (see fig. 13.5). However, to see the entire list of files in any one directory, you have to call up each file to the display, one file name at a time. To go to another subdirectory on the disk, you must choose explicitly to `Log into new disk/path`.

```
                PCSWEEP 1.5 - August 6,1985
          Copyright (c) 1985  by Koch Software Industries

                        Command Menu

F1:  Display menu.                    alt-F1:  Create a (sub-)directory.
F2:  Type current file.               alt-F2:  Find file in directory.
F3:  Tag current file.                alt-F3:  Wild card tag of files.
F4:  Untag current file.              alt-F4:  Retag # files.
F5:  Copy current file.               alt-F5:  Mass copy of tagged files.
F6:  Delete current file.             alt-F6:  Delete all [un-]tagged files.
F7:  Rename current file.             alt-F7:  Set/Reset file attributes.
F8:  Log into current subdirectory    alt-F8:  Log into new disk and path.
F9:  Save Tag parameters.             alt-F9:  Recall Tag parameters.
F10: Space remaining on disk.         alt-F10: Exit to DOS.

alt-1: Backup big file.               alt-2: Restore big file.
alt-3: Find matching files on disk.   alt-4: Find first matching file.
up-arrow:  Backup to last file.       down-arrow:  Next file.
PgUp:  Up 16.                         PgDn:  Down 16.

                   (Type any key to continue)
```

Fig. 13.4. *PCSWEEP's command screen.*

```
Volume in C: is **DOS 3.1**
Directory is C:\DOS31

    ==== 39 files occupying     242k with 10380k remaining ====

 1. .            01-01-80 23:15  <DIR>        :
 2. ..           01-01-80 23:15  <DIR>        :
 3. ANSI    .SYS 03-07-85 23:18       2k      :
 4. ASSIGN  .COM 03-07-85 23:18       2k      :
 5. ATTRIB  .EXE 03-07-85 23:18      16k      :
 6. BACKUP  .COM 03-07-85 23:18       6k      :
 7. BASICA  .COM 03-07-85 13:43      28k      :
 8. BASICPRO     01-01-80 23:15  <DIR>        :
 9. CHKDSK  .COM 03-07-85 23:18      10k      :
10. COMP    .COM 03-07-85 23:18       4k      :
11. DISKCOMP.COM 03-07-85 23:18       4k      :
12. DISKCOPY.COM 03-07-85 23:18       6k      :
13. EDLIN   .COM 03-07-85 23:18       8k      :
14. FDISK   .COM 03-07-85 23:18       8k      :
15. FIND    .EXE 03-07-85 23:18       8k      :
16. GRAFTABL.COM 03-07-85 23:18       2k      :
```

Fig. 13.5. *PCSWEEP directory.*

Once you get used to the display, you find that the program handles a number of very handy tasks. The program tags files individually or by wild-card designation (* and ?) so that you can easily execute commands on a number of unrelated files. Files can be untagged (if you make a mistake) and retagged (in case you want to repeat an operation—like copying—on the same group of files without going through the process of tagging them again).

PCSWEEP also is very much aware of the file archive bit. Any time you wish to copy a file or group of files, PCSWEEP checks to see whether the file's archive bit is set (meaning the file was created or altered since the last backup was made). Then PCSWEEP asks whether you wish to have the archive bit turned off after the copy. (You may choose to leave the archive bit on if the copy you are making is *in addition to* ordinary backup operations.) Similarly, when you use the wild-card tag, PCSWEEP asks whether you want to tag only modified files (those with the archive bit turned on). This capability facilitates the use of PCSWEEP for routine backup operations based on incremental backup concepts (see Chapters 15 and 16).

Next to the name, date, time, and size of each file, PCSWEEP's file display shows the status of all the file attributes (the archive bit, read-only, system, or hidden). You can set or reset any of the file attributes for a single file or a group of tagged files.

The program has a File Find function, which works only within the subdirectory you are logged on to, and a Global File Find, which works on the entire disk. The latter function appears to have been

added in the recent version of the program but is just too slow for for some users.

Two functions show up as modest surprises for this kind of program. First, the program comes with a file "undelete" function, which must be requested specifically either when you first start the program or when you log (or relog) on to a new disk or directory.

As described more fully in Chapter 10, when you delete a file, at first only the entry in the disk's directory is changed; the actual data in the file is not altered. Only when new files are saved to the disk is data from the deleted file overwritten. When you request the file undelete function, PCSWEEP looks for *deleted* files in the directory in addition to the usual files. The program displays the names of the deleted files, just like any other file names, except that the first character in the name of a deleted file is now a Greek sigma.

To undelete a file, you execute the Rename function, just as you would for any ordinary file. You get a warning message stating that the deleted file's data may not be retrievable but that PCSWEEP will try. After you provide a new name, PCSWEEP goes off to recover the "lost" file. If you are lucky (and particularly if you exercise this option before you have saved any new material to your disk), you have a good chance of bringing back the file as it was. Seeing this useful and easily executed option in a program like this is a pleasant surprise.

The second surprise looks like it was added in response to early user needs. The standard copy function of PCSWEEP mirrors the DOS COPY function. PCSWEEP is just as fast (or slow, depending on your point of view) as DOS, but the program cannot split files to allow for copying files larger than a diskette to multiple diskettes, as the DOS BACKUP function can (see Chapter 15).

PCSWEEP takes care of large files with a special function aptly called Copying a Big File. This function splits the file and to keep track of the division, creates a small additional file on the first disk. The reverse function, Restoring a Big File, is used to re-create the file on the hard disk if you need to restore the file from the floppy diskettes to which it was saved. You must tell PCSWEEP when the last disk of the set is reached, so you should carefully label the disks as part of a sequence (TESTFILE.DOC 1 of 2, TESTFILE.DOC 2 of 2, and so on).

The restore process is easily flustered. After the program asks whether you have more disks to process, PCSWEEP prompts you to insert the next disk and press any key to resume. Suppose that accidentally you leave the first disk of a two-disk set in the drive and

press a key. The program somehow does recognize that you have inserted the wrong disk in the sequence. But when you correct your error and signal the program to continue, it loses its place and won't resume the restoration of that file. Nothing is lost because all you have to do is start the process over again, but the process of recovering from the error should be smoother.

The program is easy to pick up and use immediately. PCSWEEP doesn't have the pizzazz of some of the current programs, but it does its basic job well. Although the user interface is somewhat primitive, PCSWEEP works nicely and even includes a couple of extra goodies as a bonus.

The Norton Commander

From the Peter Norton Company, the same company that produces the Norton Utilities, comes a hard disk file manager and modest menu system called The Norton Commander. It is one of the DOS "shells," somewhat resembling the 1 DIR program. The main screen of the program displays two side-by-side windows on the top half of the screen. The bottom half of the screen contains the familiar DOS prompt and cursor (see fig. 13.6). You still have the option of executing commands at the DOS prompt in the "old fashioned" way—typing them. Programs run as usual and then return you to the same display screen. The main screen can be configured to display either the file list and status screen or two file lists of different disks or directories.

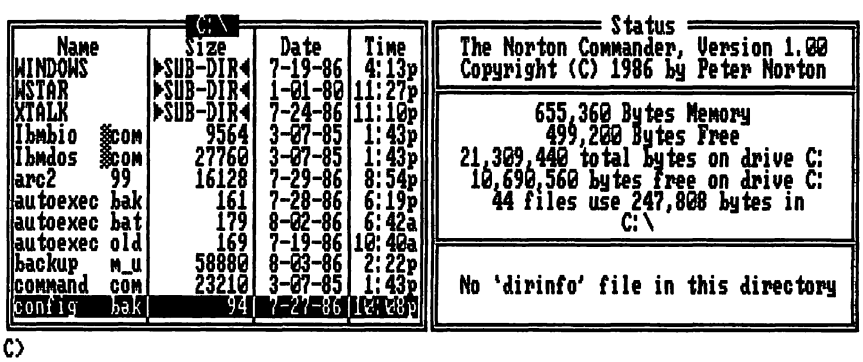


Fig. 13.6. *The Norton Commander main screen.*

You can move easily around a directory and jump to different subdirectories. To execute a program the "new" way, you move the cursor to the program name and press the Enter key ("point and shoot" is the popular term). You can access some DOS commands (like COPY, RENAME, DELETE) directly from a WordStar-like menu at the bottom of the screen.

Other commands that improve upon DOS are provided. A VIEW command takes a page-by-page look at a file; MOVE (which is cleverly described as an extension of the RENAME command) just renames the file to another disk or subdirectory; and EDIT (a small editor) serves most users' modest needs—it can edit files under 20,700 bytes. VIEW is not limited as to the size of file it can display. EDIT can even be used to look at and modify a nontext file. Hidden files also can be accessed without first requiring a change in file attributes.

When executing DOS or DOS-like commands from the menu, it is helpful to be able to tag individual files or groups of files. If you stubbornly insist on typing commands on the DOS prompt line, you still have the added convenience of on-line editing capabilities. If you enter a complicated command with assorted parameters and make an error, you can use WordStar-like editing commands to modify the line. (Although WordStar's command keystrokes are not the most memorable, Norton apparently counts on the tremendous population of microcomputer users raised on the old word processor.)

Other desirable DOS command-line functions have been added—you can recall previously entered commands, one by one, backward through the stack (with Ctrl-E) or forward (with Ctrl-X), or by typing the command's first few letters. The built-in file editor also borrows many of its command keystrokes from WordStar.

In addition to the capability to execute a program by "pointing and shooting," The Norton Commander borrows a clever idea from Digital Research's GEM™ environment manager—the ability to start an application by executing (pointing to) one of its data files.

Suppose that your word processor routinely adds the file extension *DOC* to all its processed documents. Point to one of those documents, press the Enter key, and the word processor is started. (You have to take some very minimal action beforehand to be able to do this. A special setup file must be created for The Commander to refer to.) This point-and-execute feature is another timesaver.

On top of all this, The Norton Commander lets you create your own simple menus, a different one for each subdirectory. A default menu in the root directory is activated whenever no explicit menu is

created for a subdirectory. The built-in editor can be used to create these small files that the menuing system requires.

When you run a program, The Norton Commander remains in memory and so robs some RAM from your applications program. However, the disk comes with two versions of the program. The second version can be run in a "swapping" mode, in which only a small piece of program remains in memory, just enough to recall the rest of the program after you run an application. This extra step means that running The Commander this way takes slightly longer to bring back its main screen. After trying both ways, you hardly notice the difference in time and the latter mode of operation is highly recommended. That way, you don't pull out of circulation much RAM at all (about 12,000 bytes).

You install The Norton Commander by copying a few files to any directory on your hard disk—as long as all files are copied into the same directory—and modifying your PATH command if you want to be able to call the utility from any subdirectory. The user-created menus can be specific to each subdirectory.

The Norton Commander is one of the new generation of programs, with its pop-down command boxes and mouse support. The program works fine with a mouse but is not at all handicapped without it. The Norton Commander works fast, is easy to learn, doesn't try to do too much, and is fun to use.

PathMinder

To the inveterate 1-2-3 user, PathMinder from Westlake Data Corporation looks instantly comfortable and inviting. It has the familiar Lotus-like command structure, a two-line menu with a brief explanation below it (see fig. 13.7). You execute a menu command by pressing the capital letter of the command choice or by moving the cursor to the choice and pressing the Enter key.

PathMinder has a rich collection of functions, all callable from the menus within menus. Some people may find the Lotus-like menu a bit much for a DOS shell; but, for the most part, you can easily find your way down the hierarchical menu tree. After a while, the series of single-letter abbreviations making up a complete command become second nature; but initially, learning where some of the commands are buried can require a little experimenting.

For all the power you get, PathMinder itself, like some of the other DOS shells, can be run in very little RAM. The program's

```
Run File Directory Edit Option Application Compose Log XDOS PathMinder
Run highlighted program

Filename        Ext      Size        System Status
123                     Directory
BATCH                   Directory   Sun  3 Aug 1986            4:11:59p
BOOK                    Directory
CMASTER                 Directory        Drive C Status
DOCS                    Directory   Volume Label            **DOS 3.1**
DOS31                   Directory   Bytes of Storage Total   21,309,440
  ANSI          SYS        1651     Bytes of Storage Used    10,639,360
  ASSIGN        COM        1509     Bytes of Storage Free    10,670,080
  ATTRIB        EXE       15091
  BACKUP        COM        3577          Memory Status
  BASICA        COM       27520     Bytes of RAM Total          655,360
  CHKDSK        COM        9435     Bytes of RAM Used           260,768
  COMP          COM        3664     Bytes of RAM Free           394,592
  DISKCOMP      COM        4073
  DISKCOPY      COM        4329          Log Status
  EDLIN         COM        7261     Current User
  FDISK         COM        8173     Current Account            Personal
  FIND          EXE        6403     System Log is currently         Off
  GRAFTABL      COM        1169

PathMinder v3.00 — (c) Copyright 1984, 1985, 1986 Westlake Data Corporation
```

Fig. 13.7. *The PathMinder display.*

virtual mode eats up less than 5K of RAM, leaving almost all your RAM for any applications programs you run within PathMinder. Presumably, when you move out to an application, that small chunk of program remains in memory and calls back the main portion of PathMinder from the disk when you are ready to return. Many users hesitate using the DOS shells that consume a significant portion of the machine's RAM because they would rather have it available for applications programs. The virtual mode of PathMinder makes it much more palatable—indeed tempting—to use.

The other operational mode of PathMinder—the resident mode—keeps the program totally in RAM while you run an application. Resident mode consumes about 115K of RAM. The advantage is that the shell snaps back more quickly because it is all in memory, waiting to bounce back into action. The difference in time, happily, doesn't make the loss of RAM a reasonable tradeoff. For the three-second loss, choose the virtual mode.

Applications can be called up and executed in several different ways. You can highlight the program in the file display window and select the first menu option, RUN. Depending on how you set up RUN's options beforehand, you can have PathMinder prompt you for any parameters the applications program might require. The second way of running an application is from the Application Menu, a user-defined menu that is structured just like the rest of PathMinder's

menus. The third way of running an application from within PathMinder is the "if all else fails" mode—the COMPOSE command. You type the command as you would at the DOS level. The directory from which the command is executed is determined by the file or directory currently highlighted in the file display window.

The Application Menu creation process is a well thought out feature in its own right. This handy and flexible feature (called AutoMagic) guides you through the steps involved in building Lotus-style menus like PathMinder's own. Yes, the menu creation is menu driven! A menu can have a mix of applications and submenus, in true hierarchical fashion, to mimic any logical application structure you have in mind. You can provide preset parameters or prompt the user for them at run time. For instance, you can set up the menu so that the user highlights a word-processing document, calls up the Application Menu, and selects the desired word processor. PathMinder feeds the document to the word processor as it is firing it up.

Obviously, much careful thought was put into the design of many parts of the program, including the Application Menu. Here and there you can notice some "Nice Little Touches." In the menu-creation process, one striking design consideration allows for a menu option to be executed by the first *capital* letter in the option name rather than simply the first letter in the option. Suppose that you have two applications in the same menu, one called Wordstuff and the other called Wordtime. You can type these names into the menu creation step as *wordStuff* and *wordTime* and start wordStuff with *S* and wordTime with *T*.

A couple of other nicely done features round out PathMinder. One feature is a file encryption/decryption feature, which you can use to safeguard data or program files from prying eyes. You have to be quite careful with any sort of file-encryption program. If you forget the password used to scramble the file, you are very much out of luck. The documentation emphatically warns that even Westlake cannot recover your data.

Do not take chances encrypting copy-protected programs. Again, the documentation gives appropriate warning that, because of unusual copy-protection schemes, you may not be able to run the software at all after encryption. PathMinder does include a nice safety feature: you can choose to have the original (unencrypted) file retained. This feature is a good idea; you can keep the original safe (just in case the encryption messes up any copy protection) and try the encrypted copy out in general circulation.

PathMinder also comes with a system log function, which you can set up to keep track of computer usage by user and by project. The function stores on disk a usage log that records details like time logged in, each command or application run, time devoted to each, and time logged off. This function is not meant to be Big Brother watching over your shoulder but rather a convenience to document usage. The function could be handy for documenting the varied usage of a departmental machine, for example.

When the program is first installed, the automatic installation substitutes (and politely renames without destroying) its own model for your own AUTOEXEC.BAT file. PathMinder's AUTOEXEC.BAT includes helpful hints on how you can modify it to suit your own needs. You also can bypass automatic installation.

PathMinder's documentation is an excellent model of good documentation. It is well organized and very clearly lays out the options and explains them with examples. The documentation does its job well and is a pleasure to use, and the program itself reflects this clarity.

PCTools

PCTools is a very useful though not slick-looking utility produced by Central Point Software, Inc. PCTools has many of the features of the other hard disk managers discussed here plus a few unique ones. When you first boot up the program, the screen looks so crowded that less hardy souls may be discouraged. After exploring the utility a bit, though, you quickly recover from your first impression.

Whereas many other hard disk managers act as DOS shells, PCTools takes a different tack. This program is more like a DOS "seed." You don't run other programs from PCTools; you run it from other programs. PCTools leaves your ordinary operating procedure alone but waits in the background until you need it. You also can run PCTools from the command line as a stand-alone program.

When you call up PCTools, after an introductory screen (or whenever you choose to change the currently displayed disk or directory), the program displays a simple directory tree (see fig. 13.8). On this tree, you move the cursor to highlight the branch you want to see. Then the screen is split into three parts (figure 13.9). The main part of the display is the file list (for the selected subdirectory) in the middle. At the bottom is the menu from which you select commands by a key letter, and just above that is a count of files/bytes available and marked or selected.

```
                                                          **DOS 3.1**
Path=C:\                                                  BLINK=DOS current

C:\
 ├─WINDOWS──────PIF
 ├─UTILS
 ├─DOS31────────BASICPRO
 ├─BOOK
 ├─DOCS
 ├─BATCH
 ├─WSTAR
 ├─SYMPH10
 ├─MM
 ├─MACE
 ├─NORTON
 ├─WP
 ├─SUPER4
 ├─DV
 ├─SUPER3
 ├─TOPVIEW
 ├─LIGHT
 ▼
```

Fig. 13.8. *The PCTools directory tree.*

```
PC Tools R2.02                                            Vol Label=**DOS 3.1**
                                  File Functions                  Scroll Lock OFF
Path=C:\UTILS\*.*
   Name     Ext    Size Attr   Date        Name     Ext    Size Attr   Date
   C-WHIZ   EXE   49792 ....  2/27/85      QUERY    COM      99 ....  5/25/86
   ASK      BAT     252 ....  5/25/86      SETCLOCK COM     853 ....  9/19/82
   ASTCLOCK COM     813 ....  9/18/82      SHIPDISK EXE   17478 ....  1/16/86
   BLANK    COM     264 ....  9/16/84      SPOOL    BAT     157 ....  5/04/86
   CHK4BOMB ARC   11648 ....  6/09/86      SUPERSPL COM    6021 ....  4/12/83
   DOS-WS   BAS    1865 ....  9/23/84      VTREE    COM     512 ....  4/20/86
   DPATH20  COM    2560 .... 10/23/85      WHEREIS  COM     512 ....  8/28/84
   DPATH20  DOC    3060 .... 10/23/85      WS-DOS   EXE   22784 .... 10/19/84
   DPROTECT ARC    3584 ....  6/09/86      XMAKER   BAS     670 ....  9/16/84
   KEYBUFS  COM    1152 ....  4/20/86      VIEW     COM    2688 ....  8/04/84
   KEYBUFS  DOC    2816 ....  4/20/86      FORMAT   BAT     421 .... 10/29/85
   MGC      COM    6784 ....  1/01/80      TYPE             0 ....  7/16/86
   PC-SWEEP ARC   38272 ....  4/20/86      MENUHD   BAK     778 ....  3/13/86

  62 files LISTed   =   898150 bytes.   62 files in sub-dir =   898150 bytes.
   0 files SELECTed =        0 bytes.   Available on volume = 10641408 bytes.

  Copy Move Compare Find Rename Delete Verify view/Edit Attribute Print List
Sort Help F1-SELECT F2-UNselect F3-alt dir lst F4-other menu ESC-exit PC Tools
  F5-directory LIST argument  F6-File SELECTion argument  F7-chg drive/path
```

Fig. 13.9. *The PCTools split-screen display.*

At first, the command menu may look a little crowded. In addition to the equivalents of the DOS COPY, RENAME, and DELETE commands, the program has built in another handy command—COMP (to compare two files byte by byte to see whether they are exactly the same). PCTools works like many of the file managers in that you can mark or tag a group of files for certain commands to act on.

Interestingly enough, PCTools' DOS-like commands seem to run more slowly than commands in DOS. Having COMP available is good, for example, but the command is about half as fast as DOS COMP. The big advantage with PCTools is that you can tag or mark only the files you want compared. DOS forces you to specify the files in a single file specification (including wild-card characters).

Each time you call up the program (either directly or from the background RAM-resident mode) PCTools makes you wait while it grabs a snapshot of your present hard-disk directory structure. If you need to make a copy of one single file, for example, the wait is just long enough that you may prefer to exit from your application and do the copy at the DOS level.

PCTools has some extra goodies that DOS does not provide. The program has a full-screen editor (in pleasant contrast to DOS's line-oriented editor EDLIN). This editor lets you view files in plain ASCII text or hexadecimal form. The FIND command helps you find the occurrence of any string of characters in a file on the disk. You can easily change the attributes of any file (read-only, archive, system, or hidden), and—a nice touch—you can even change the date and time recorded for the file in its directory. The way PCTools prompts you is also very pleasant (see fig. 13.10). A brief help screen comes up so that you can move the cursor through each attribute rather than type each one you want.

By the way, when you operate on files within PCTools, the program doesn't respect the read-only, system, or hidden file attributes (other than in the attribute display). In other words, if you select a group of files to delete, PCTools wipes them out (after confirmation, of course) even if some are hidden, system, or read-only.

You have options that sort and print a directory many different ways, and you can undelete a file. Everything is prompted nicely. The program has a separate disk/directory command menu (see fig. 13.11), which offers many of the file commands—COPY, COMPARE, FIND, VERIFY, VIEW/EDIT, UNDELETE—now applied at the disk level.

```
Path=C:\UTILS

File=KEYBUFS.COM        Size =    1152 bytes
                        #Clu =       1 clusters

          Initial attributes shown indicate those in effect. To
          change attributes, use the arrows (↑↓) to select an
          attribute to change. ENTER (↵) will change (toggle) that
          attribute. Pressing "U" will update and make the changes
          permanent. "ESC" will return without any change.

          Initial Attributes      New Attributes
           Read Only - OFF         Read Only - OFF
           Hidden    - OFF         Hidden    - OFF
           System    - OFF         System    - OFF
           Archive   - OFF         Archive   - OFF

          Initial Time/Date       New Time/Date
           11:16p                  11:16p
           04/20/86                04/20/86
```

Fig. 13.10. *A PCTools help screen.*

```
PC Tools R2.02
                    Disk and Special Functions

DISK SERVICES: Copy cOmpare Find Rename Verify view/Edit Map Locate iNitialize
SPECIAL SERVICES: Directory maint Undelete system Info Help
  F3=return to file services  Esc=exit PC Tools
```

Fig. 13.11. *The PCTools Disk-directory command menu.*

The MAPPING option displays a map of the hard disk, showing occupied, available, and bad clusters as well as the location of hidden, system, and read-only files. In addition, the Special Services option includes an extensive subdirectory maintenance function, which lets you rename, create, and remove subdirectories.

PCTools has another special feature called Prune and Graft, which allows for a more freewheeling manipulation of the subdirectory structure. For example, you can move a subdirectory (with all *its* subdirectories) to an entirely different branch of your hard disk tree.

The only apparent problem when running the program RAM-resident is probably the result of the way the program is used. Suppose that you are working in WordStar and call up the program by pressing Ctrl-Esc. Suppose also that when you use WordStar, you work with a couple of "phony" disk drives created with the SUBST command (see Chapter 9). When you choose to operate on a file located on one of the alias disk drives, PCTools says that it can't find the file even though it shows up in the file list.

This program could use a little improvement in its screen organization and layout. You do have to be careful not to "play with" a file on which your applications program is working. This is probably one reason that PCTools has no equivalent to CHKDSK. (You don't want to be running CHKDSK with the F option, fixing up the disk while you are still running an application.)

The DOS "seed" concept (the ability to call up PCTools from within another application) is very good. If you feel frustrated by the constant necessity to exit from an application to take care of some disk or file management function, PCTools may be the tool for you.

Public Domain and Shareware Programs

In the public domain or under the shareware concept are many programs that also fall under the headings of either hard disk manager or hard disk menuing system. Some of them are simple menuing systems only, and others help you out with various DOS commands. More information about these programs is given in Appendix B.

AUTOMENU

One shareware program is called AUTOMENU or some variation on that name. AUTOMENU is a fairly nice-looking menuing system for a hard disk.

AUTOMENU contains a series of either "pages" or pull-down windows that display groups of related menu items (see fig. 13.12). For instance, one window might contain your BASIC programs, another could have your main applications programs, and still another could contain the set of commonly used DOS commands. As you move the cursor left and right, the display changes to the different menu pages or windows. Moving the cursor up and down steps through the different choices within a menu. Pressing the Enter key executes the choice on which the cursor is resting.

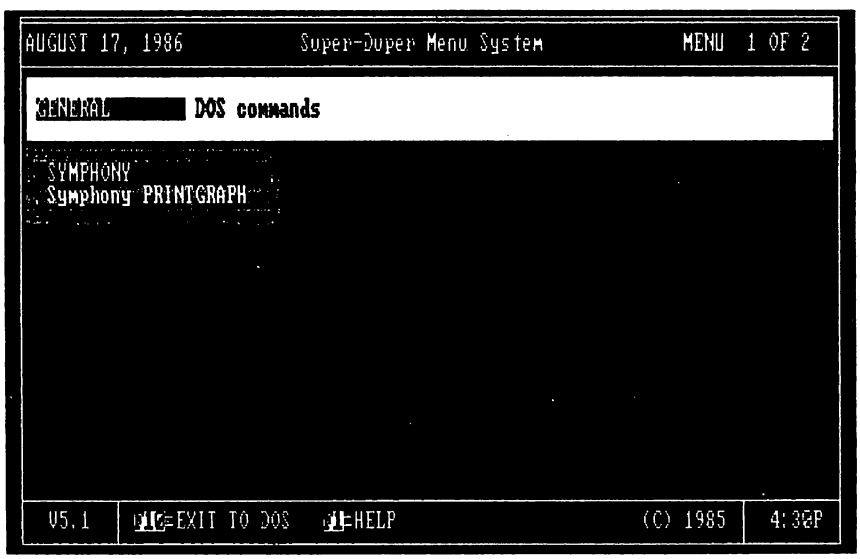


Fig. 13.12. *An AUTOMENU pull-down window.*

The display shows the date, the time, a title, and an indication of what menu page is displayed. A one-page help screen provides some modest assistance. The model even blanks out the screen if no keypress has occurred for several minutes. This feature helps preserve the screen, which some folks believe can have an image literally burned into it if left on for hours with no changes.

AUTOMENU is the ultimate stripped-down menuing system, but the program is adequate for some purposes. It has no menu-creation module, however. With your own text editor or word processor—none is provided—you must create a text file, organized according to conventions described in the program "documentation" (itself usually contained in a file provided when you somehow acquire the program).

For each option, you place into this text file the same set of instructions you would use in a batch file to execute the option. At the end of each option's set of instructions, you insert the MENU statement, which calls up the menu again. In this way, when you exit from the executed application, the menu pops up again. The program is basic but not bad once you get the hang of it. (AUTOMENU is also available in a souped-up commercial version which adds more features and ease-of-use.)

G-WHIZ

One of the handiest shareware utilities falls under the loosely defined category of file manager. G-WHIZ, whose name probably follows in the tradition of the programs GCOPY (Global Copy) and GDEL (Global Delete) designed by the same programmers, handles four DOS and DOS-like functions in a very neat and convenient fashion.

In a nutshell, you can copy, delete, move, or print a group of files by selecting (tagging) their names from a displayed list. When you start the program and get past the title screen, you are first asked to indicate which function you want to execute (see fig. 13.13). Then if you haven't already specified a source in the command line calling up the program, you are asked to enter the source path name and file specifier. For example, if you want to choose from every file in the \DOCS subdirectory on drive C, you type

C:\DOCS*.*

If you want only files with extensions of BAK, you type

C:\DOCS*.BAK

For a COPY or MOVE operation, you also indicate `To which directory or drive do you want to copy these files?`

The program then shows a full-screen display of all files satisfying the specified file selection (see fig. 13.14). You can mark any file you want included in the operation by moving the cursor to the file name and pressing any alphabetic key (the space bar unmarks a file if you make a mistake). Press the Enter key and G-WHIZ does the rest.

The function keys offer some help. One sorts the file list by root name, another sorts by extension, and another reports the total number of bytes included in the marked files (a handy statistic to know if you are copying to a limited capacity floppy diskette).

G-WHIZ includes the DOS-like but DOS-missing function MOVE. The only DOS way to do this is to copy a file and then erase it from

```
G - W H I Z                                  Copyright (c) 1985, Gordon Waite

Choose one of the following functions by using the cursor keys to move
the hi-lite bar over your selection, then pressing the RETURN key...

   Copy the marked files to another directory or drive.

   Move the marked files to another directory or drive.

   Print the marked files.

   Delete the marked files.


   Exit from G-WHIZ without doing anything to the marked files.

There are two HELP SCREENS available; press F1 or Alt-F1 to view them!
```

Fig. 13.13. *G-WHIZ option screen.*

```
G-WHIZ    Copyright (c) 1985, Gordon Waite    All Rights Reserved!

ANSI.SYS          MODE.COM
ASSIGN.COM        MORE.COM
ATTRIB.EXE        PRINT.COM
BACKUP.COM        RECOVER.COM
CHKDSK.COM        RESTORE.COM
COMP.COM          SELECT.COM
DISKCOMP.COM      SHARE.EXE
DISKCOPY.COM      SORT.EXE
EDLIN.COM         SUBST.EXE
FDISK.COM         SYS.COM
FIND.EXE          TREE.COM
XFORMAT.COM       VDISK.SYS
GRAFTABL.COM      BASICA.COM
GRAPHICS.COM      IDRIVE.SYS
JOIN.EXE          RTRTR.BAT
KEYBFR.COM
KEYBGR.COM
KEYBIT.COM
KEYBSP.COM
KEYBUK.COM
LABEL.COM

Move w/CURSOR keys, Mark w/ALPHA keys, then RETURN; F1=HELP F3,F4=SORT Esc=EXIT
```

Fig. 13.14. *G-WHIZ displayed list of file names.*

the source directory. MOVE is much faster (when moving between directories on the same disk) because it changes just the directory entry for the file. All in all, G-WHIZ is a handy utility.

Summary

This chapter reviews a few of the many hard disk file managers and public domain programs. From this information you can decide each program's worth to you. The discussion also points out advantages and weaknesses in each program. You can apply this information to other programs you may be thinking about purchasing. Some of the main points to be considered are

- Hard disk management software generally falls into three categories: menu-development programs, file managers, and program environments.
- Different file management programs provide different capabilities and also require different amounts of RAM. Consider each in the light of your own computing resources and needs.
- Programs like FDO help you create hierarchical menu systems. With FDO you also can create many different and specific prompts.
- Programs like 1 DIR create a shell around DOS but still leave you free access to DOS. With 1 DIR you can create complex menus and command sequences.
- XTREE helps you manage directories and files on your hard disk. XTREE also can be used for backing up your files.
- PCSWEEP provides file management and provision for copying very large files. This program also has a file undelete function.
- The Norton Commander provides a DOS shell and improves on many DOS capabilities. You can also start an application by pointing to one of its data files.
- PathMinder, a file manager organized like 1-2-3, helps you build complex menu systems. PathMinder uses only 5K of RAM in its virtual mode.
- PCTools is more like a DOS "seed." You run it from other programs rather than run other programs from it. You can take care of file management functions without exiting from your application.
- Shareware programs, like AUTOMENU and G-WHIZ, are another source of hard-disk menuing and file-management programs.

Chapter 14

Operating Environments

Hard disk management software covers a broad range. The programs discussed in the preceding chapter include menu development tools that help you create hierarchical menus to guide users through the various applications and DOS commands, and programs that deal primarily with hard disk file management. This chapter explains a third area of software products: *enhanced environments*. TopView®, Windows, DESQview, GEM™, and similar programs contain different combinations of file management tools and other more advanced capabilities.

As in the preceding chapter, only a few of the products in this third category of hard disk management software are reviewed. From the discussion you can form your own conclusions about these products and learn what to look for when purchasing other packages.

The products in the third category of software expand the idea of the DOS shell into the DOS *enhanced environment*, in which DOS is not only surrounded but nearly enveloped. In addition to file management capabilities, this group provides more complex features like windowing, switching between programs, transferring data between programs (cutting and pasting), and multi-tasking. The whole idea is to turn the computer screen into a computerized "desktop," on which to open, run, and close applications.

Applications developers can take advantage of the structure of the enhanced operating environments. Developers can write programs that are tailored specifically for the features of individual environments. The idea of having a uniform interface with the user is appealing to many—the users, the applications developers, and—not least of all—the creators of the operating environments.

This widespread appeal is why some of the major forces in the software industry (and some of the lesser greats) are slugging it out. At this writing, the major competition in operating environments is among IBM's TopView, Microsoft's Windows, Quarterdeck Office Systems' DESQview, and Digital Research's GEM Desktop.

Much of the attractiveness of these environments depends on your particular collection of hardware. Although some of the environments require as little as 256K, the program's abilities are heavily constrained. Similarly, some of these programs claim to run on a floppy disk system. Although they may literally *run*, the convenience of the environment is wiped out by the floppy-diskette shuffle required.

If you really want to have two or more programs "instantly" available to switch between, you will probably need a full complement of RAM in your computer. For PCs and XTs, that means 640K. In an AT, that means 640K of basic RAM and maybe even extended or expanded memory stretching into the megabytes.

Your choice of operating environment may be limited by the kind of display adapter you have. GEM and Windows require some kind of graphics display—for example, the standard IBM Color Graphics Adapter (CGA), the IBM Enhanced Graphics Adapter (EGA), or some compatible adapter. TopView and DESQview run on virtually any display device although the fact that they are not graphics-based is seen by many as a distinct disadvantage. GEM and Windows run in graphics mode, which is tolerable on a CGA and exquisite on an EGA. GEM and Windows can use icons, those little pictures of disk drives and application symbols that you point at to activate, but DESQview and TopView are strictly menu-based.

You should try out any of these environments cautiously. Some of them really flex your microcomputer's muscle, and they may wind up tripping over idiosyncrasies in your applications programs (or your mode of operation) that have not been anticipated. The developers can't test an environment with every possible combination of software.

You may find that particular combination of events that brings your system crashing to a halt. (Well, just say it comes to a halt,

which means that your screen freezes or fills up with those funny-looking characters, or that the keyboard no longer seems attached to the computer.)

As long as you are experimenting or saving your work frequently, a system crash is no disaster. You may even have experienced a crash while running a program under very uncomplicated circumstances. In any case, you need only to reset the computer, either by pressing the Ctrl-Alt-Del keys at the same time (for a "soft" crash) or, if that has no effect, turning your machine off and on (for a "hard" crash).

By the way, if you have to turn the machine off, you should wait a few seconds for the hard disk to settle down before you flip the machine back on again. About 10 seconds on a Personal Computer XT and 30 seconds on an AT is about right. COMPAQs, which are notorious for slow power supply recycling, may require 30 seconds.

GEM Desktop

The GEM Desktop (*G*raphics *E*nvironmental *M*anager) from Digital Research brings to the IBM personal computer family a Macintosh™-like environment. GEM features a menu bar at the top of the screen with pull-down submenus; icons, or symbolic pictures representing disk drives, subdirectories, files, and applications; and the representation of the desktop itself (see fig. 14.1). With GEM acting as a front-end menuing system for DOS, you can manage your files on a hard disk; and you can call up applications programs from within GEM Desktop.

GEM works much better with a mouse although you can get along using just the keyboard. Every mouse operation has its equivalent on the keyboard, but using a mouse to navigate around the desktop is quicker and more convenient. With a mouse or not, you need to learn the mouse lingo: clicking, double-clicking, dragging, and shift-clicking.

Clicking is a press and immediate release of the mouse button; *double-clicking* is two quick presses and immediate releases. To *drag*, you hold down the mouse button, move the mouse, and release the mouse button. *Shift-clicking* is clicking on several icons while holding down the Shift key and then releasing the Shift key. This technique is used to select multiple icons at one time.

When you double-click with the cursor on the hard disk icon, more icons fill a window on your screen desktop. These icons represent each data file, applications program, and subdirectory in

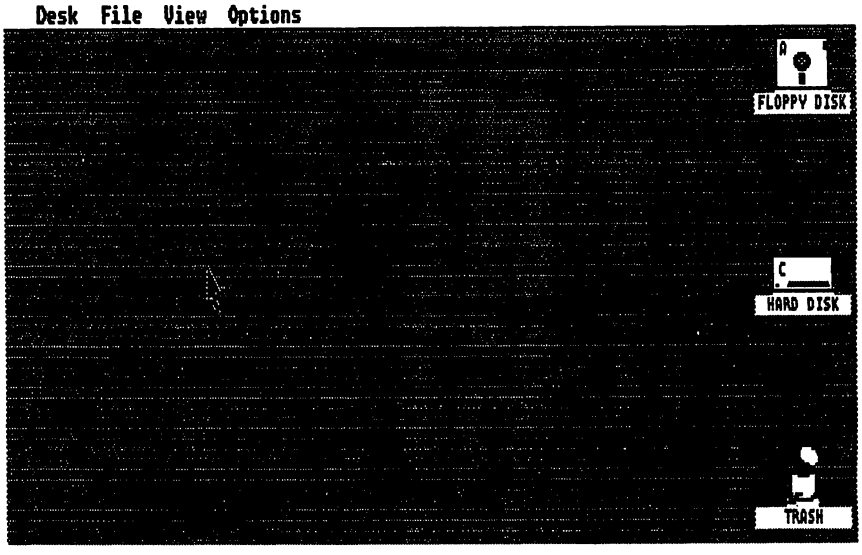


Fig. 14.1. *The GEM Desktop.*

the root directory. Double-click on a subdirectory, and out spray its files and subdirectories into another window, smaller than the first window and laid on top of it (see fig. 14.2).

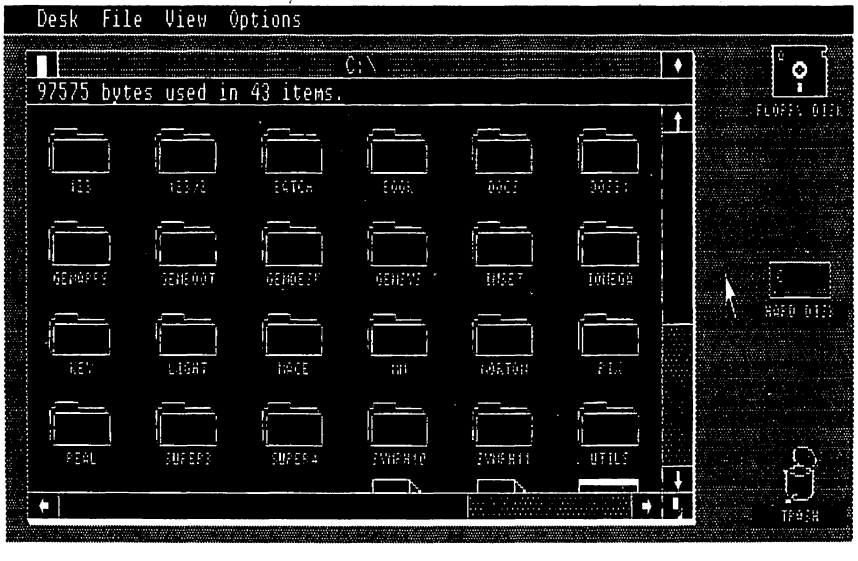


Fig. 14.2. *The GEM Desktop icons.*

If you don't like the icons, you can select a more standard directory display (see fig. 14.3). If you like the icon idea but want variety, the program has a set of more distinctive icons from which you can choose when you install an application.

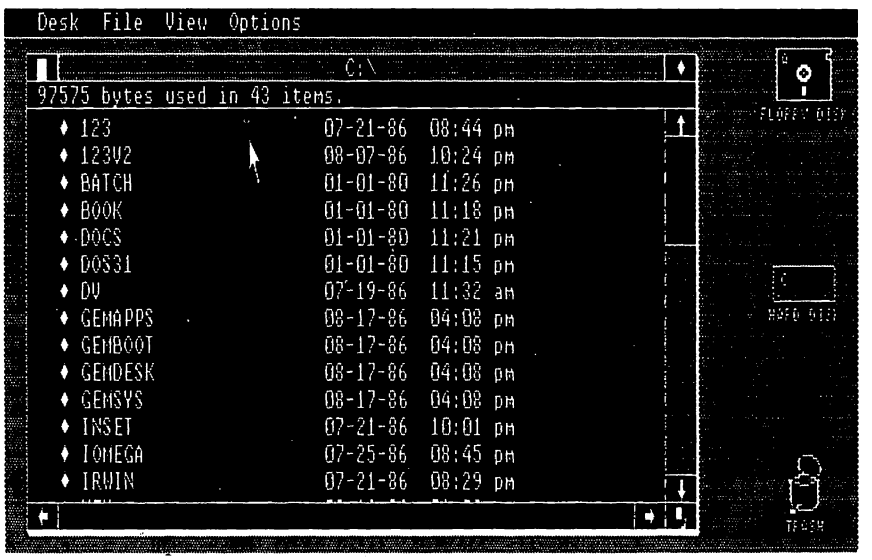


Fig. 14.3. *The GEM Desktop directory display.*

If the window is not large enough to show the entire display of icons or text lines, you can scroll the text within the window, both horizontally and vertically. Once you find that you prefer a particular layout of windows, you can save the entire desktop layout so that every time you start GEM, the same assortment of windows pops up.

When you select any directory icon, you can display the creation date and time and the byte size as well as whether the file is read-only. Although DOS allows you to change the read-only attribute only with Versions 3.0 and above, GEM lets you change that attribute with lower versions (V2.0, V2.1).

When you double-click on an application (which to GEM is any icon or text line that is not a subdirectory), GEM tries to execute the program immediately. GEM stays resident in the background and pops its desktop back on the screen after you finish running the application.

From within GEM, you can create subdirectories, navigate easily through the path of directories (watching each subdirectory display

its contents); copy files, groups of files, or whole disks; rename files; and format disks.

Some desk "accessories" are available: pop-up calculator, clock, and print spooler. The print-spooler function allows you to tell GEM to send a group of files to the printer. The files can be printed immediately or remain in the spooler while you return to other tasks. (Print spooling can be used only for text documents or documents created by GEM applications like GEM Draw, GEM Paint, or GEM Write.)

In case you want to do something that GEM has not specifically provided, you can move temporarily into a DOS window, execute any DOS commands or applications on your own, move back to GEM, and pick up right where you left off.

You can, in some cases, execute an application by opening a data file belonging to that application. This capability avoids the need to open the application and then select the document or data file you need. You must set the appropriate parameters beforehand, however. This feature is restricted mostly to GEM applications (GEM Write, GEM Draw, and GEM Paint, as of this writing) and word processors that save their files with one required file extension (DOC or TXT, for example).

GEM is not copy-protected and comes with its own installation program, which handles the entire setup process. Only two minor complaints can be voiced about the hard disk installation program. Because GEM uses the DOS MODE and FORMAT commands, GEM's install program automatically puts those two DOS programs in your root directory, whether or not the files already exist somewhere else on your hard disk.

For some reason the install program also asks you to insert a *floppy diskette* to supply the FORMAT command, which should be unnecessary when FORMAT is already installed in a PATH directory. The same is true for MODE. Assuming that your active PATH command has specified the subdirectory containing those commands, removing those extra files from the root directory is recommended. If you are keeping all batch files in a separate subdirectory, you also may want to move the start-up batch file created by the install program.

A minor quibble with the program operation itself arises when copying files. When GEM encounters a file name conflict (for example, you try to copy TEST.DAT from one directory into another that already has a file by that name) or when GEM is ready to go ahead with the copying, the program displays a small box and asks

you to verify the operation. Verification is a good idea, but the program doesn't provide the path of the file you are copying from or to. Seeing that information before going ahead with the copy would be helpful.

GEM takes up only about 45K of RAM. As a simple graphics-based hard disk file manager, GEM does a very good job. If you like mice and icons, GEM Desktop might be for you.

Windows

When you first start Microsoft's Windows, the initial shell program—the MS-DOS Executive—appears on the screen (see fig. 14.4). This program is the desktop manager that guides your session. At the top of the window, which fills most of the screen, are icons representing your disk configuration—floppy disk drives and hard disk drives. The window also shows the current directory path and list of files in that subdirectory. The program assists you in some of the basic file management operations: copying, renaming, deleting, and printing files; changing, creating, and sorting subdirectories; and formatting a data or system disk, all from within the Executive.

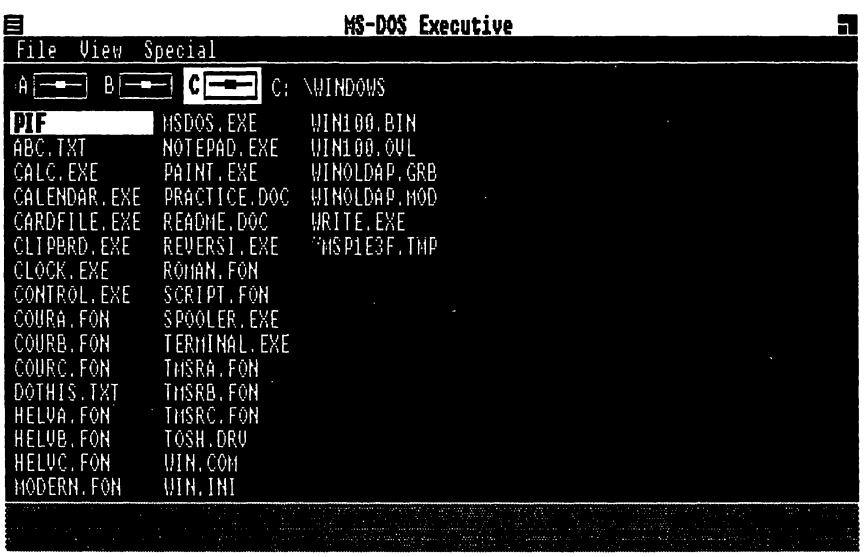


Fig. 14.4. *The MS-DOS Executive.*

You can also run an application by just selecting the program file name. (Depending on whether you use a mouse or the keyboard, the method of selecting varies. You may find that the mouse lingo—clicking, dragging, and so on—is used in this description because that was the version of Windows used.) You can move down and up the subdirectory tree by pointing to the appropriate subdirectory or parent directory. Using the mouse with Windows appears to be the easiest method. The capability to drag those little icons and move them around the screen as well as point and click on the menu choices and file (application) names is a big plus.

Installation is a snap. Windows itself comes on four diskettes, and the word processor is on a fifth diskette. To install the program, all you do is place the first diskette in the drive and type **SETUP**. The program prompts you to insert the other diskettes when needed, creates a WINDOWS subdirectory on your hard disk, and copies about 100 files into that subdirectory. The program very politely leaves your AUTOEXEC.BAT and CONFIG.SYS files alone. Any change is left up to you.

If you decide to run a data file, the Executive first determines whether the file is a word-processing document belonging to Microsoft WRITE or a graphics document belonging to PAINT (both programs provided with Windows). The Executive starts the appropriate program and brings in the data file ready to be edited.

The Executive comes with a collection of desktop utilities. It has a calculator, note pad, clipboard, and clock. (Seeing the old-fashioned clock hands, complete with second hand, is a bit unsettling.) The clipboard actually represents a buffer area in memory. The buffer is used when transferring information from one part of a document to another or one window or application to another.

Multiple windows can be created, expanded, contracted, and moved around. Microsoft distinguishes Windows by using the concept of *tiling*, in which the windows are shuffled and resized so as never to overlap (see fig. 14.5). When you are finished with an applications window, you can zap it. The window disappears from the screen, and the remaining windows are resized and rearranged.

You can temporarily close a window by dragging it to the bottom of your screen and parking it there. (Dragging turns the window into an icon.) The application stays in memory (or gets swapped out to expanded memory or disk if necessary), and the icon remains at the bottom of the screen, ready to be activated at the punch of a mouse button.

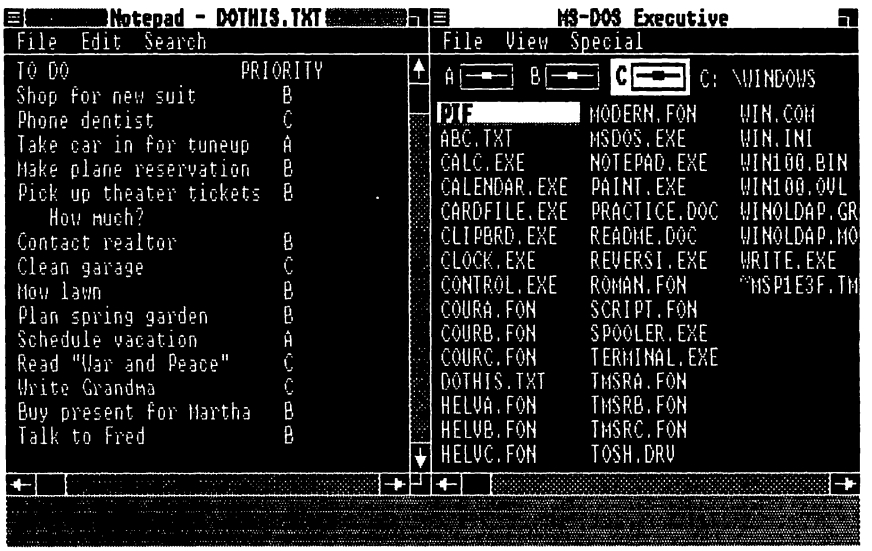


Fig. 14.5. *Microsoft Windows tiling.*

Obviously, your machine's memory can hold just so many applications at a time. Depending on the application, Windows may swap it out to disk, thus freeing up memory for other applications to run. When the swapped application is called again, Windows brings the application back exactly as it was before you left it.

The amount of extra RAM, extended memory, expanded memory, and hard disk space determines how quick this process is. The speed of your computer's microprocessor chip has a major bearing, too. On a plain old PC or XT with a floppy disk drive, the processor cannot really carry the load well. You need an AT (or an accelerated PC XT processor) unless you are very patient.

The graphics-based environment of Windows also slows it down compared to a text-based environment. Everything, even each character in a word-processing document, is created in graphics mode. Of course, that factor also gives you a "what you see is what you get" capability in an application like a word processor. Underlining shows up as underlining, not control characters, like ^U, before and after the underlined phrase. The same goes for other display characteristics—italics and bold face—and different type sizes and fonts.

Certain applications, which are not so-called well-behaved programs, ordinarily cannot function in many windowing

environments, including Windows. These misbehaving programs are written in a way that bypasses standard DOS conventions. The authors of these programs took this route as a way of speeding up the programs. 1-2-3 and WordStar are prime examples of this.

For many windowing programs, these and other popular applications may not run within a window but can be run in full-screen mode. In this mode, the application takes up the full screen and looks just like it looks when you are running it outside the windowing environment. The difference is that you are still running somewhat within the environment; after exiting from the application, you automatically pop back to Windows. Also, in the case of Windows, you still can transfer information (in text form) from the misbehaving program into an application running within a Windows window. (That's what you use the clipboard utility for.)

In order to use any application not specifically designed to work with Windows, the program must have certain information about the application. Otherwise, Windows uses its default settings, which may not be good enough for the application to work efficiently—or at all. The file information is stored in a specially formatted file called a *program information file*, or PIF.

A few dozen PIFs are provided on one of the Windows disks. Because Microsoft would like to establish Windows as the standard operating environment, the company has made the data requirements and file specifications readily available to developers. As a result, you may find more and more applications packages including a PIF file to facilitate installation under Windows.

Although Windows can manage more than one application at a time, all *active* applications are not really active. Only one application is active—the one whose window is active. The others assume a holding pattern. When you jump back to a particular window, its application continues to execute.

Windows is very slick looking. Its speed of operation on a PC or XT may be a bit disappointing but becomes more tolerable with some alternative configurations. In recognition of the problem, Microsoft is marketing Mach10™, a PC speed-up board that replaces the PC XT's 8088 microprocessor chip with a swifter AT-like 80286. (Mach10 is not the first or only such board on the market, but Mach10 is the first one from Microsoft.) Windows may appeal to users who need constantly to go back and forth between a few applications.

TopView

The debate rages on as to whether IBM is trying to shove TopView down users' throats as a required top end to its future operating systems. Regardless, IBM's presence in the industry forces users at least to give some consideration to TopView.

Although most programs can run under the TopView environment, not all can take advantage of its potential. In relation to TopView, software falls into one of three categories: TopView-specific programs, well-behaved programs, and not-so-well-behaved programs. TopView-specific programs have been written (or rewritten) to conform to TopView specifications. Well-behaved programs are ones that generally conform to standard DOS conventions and don't bypass DOS during input/output operations. Not-so-well-behaved programs can run in TopView but with definite limitations.

Both TopView-specific and well-behaved programs run in windows that you can move around, resize, and hide. These programs also run in a multi-tasking mode. You can actually have more than one program executing simultaneously. Although the not-so-well-behaved programs can run within TopView, they cannot run in a window (they take over the full screen), and they can run only in the foreground. When you switch to a different program, execution freezes until you switch back to the first program.

If TopView (Version 1.1) needs more memory to run a new application than the currently running applications leave, TopView swaps one of the running programs to extended memory, expanded memory, or disk (either a real disk or a RAM disk) and uses the released space for the incoming application. Of course, the execution of the swapped-out program is frozen. When you switch back to the swapped-out program, TopView reloads the application and picks up processing where it left off. Your menu of open applications indicates whether a program is running, swapped out to disk, running but with its windows temporarily hidden (at your request), or temporarily suspended. (You might choose the last option for a program you wish to suspend until you can devote undivided attention to it.) You do have to be careful what programs you let TopView swap. Communications programs, for example, don't communicate well from a frozen position. You can prohibit TopView from swapping selected programs by setting the appropriate parameters in the program information file.

TopView's controlling menu (see fig. 14.6) is callable by the press of a button (the Alt key or a mouse button). Your choices at any given time are indicated by a small square to the left of the menu item. For instance, if you are running an application that can use only a full screen, the Window menu choice is marked as unavailable for your use.

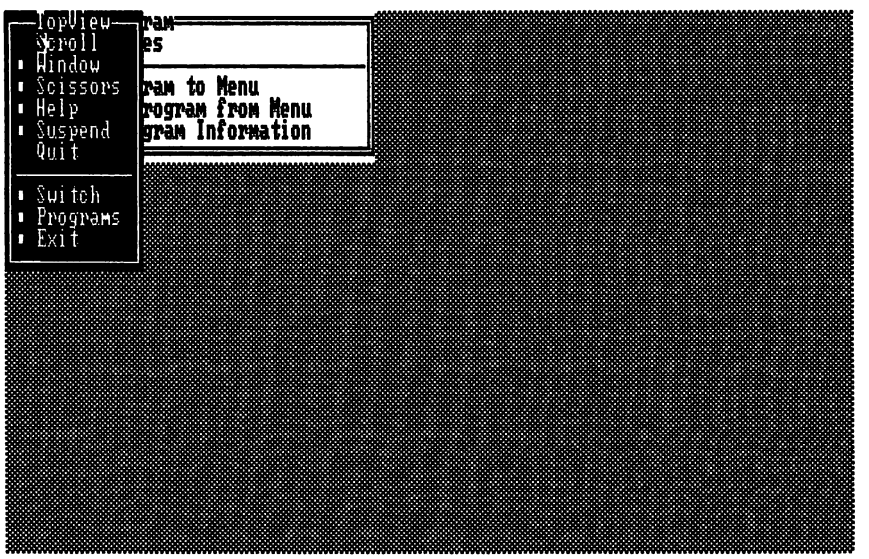


Fig. 14.6. *TopView's controlling menu.*

One of the exciting features of enhanced environments is the promise of application-to-application data transfer, commonly referred to as the cut-and-paste operation. TopView is set up to allow for this; but unfortunately, except for a few special programs, only cutting (indicating material to be copied) seems to work, not pasting (the actual transfer of information *into* the application). When you start to paste, the menu doesn't even allow a paste or transfer option. At this writing, only the TopView-specific programs can paste information. The cut-and-paste operation is wasted for any programs that are not TopView-specific.

How does TopView know how to treat each program and how to handle its idiosyncrasies? To run a program efficiently (or at all), TopView gets the necessary operating parameters about each application from a Program Information File (PIF). The PIF has an IBM-defined structure to hold a specific set of parameters. PIFs for a

limited assortment of IBM-produced programs come with TopView. Some other applications packages include a PIF file to simplify the program's use under TopView.

IBM now includes a separate Application Guide that lists a wide assortment of programs and their PIF parameters. You can then create the PIF under TopView, filling in the relevant operating information. Even for programs not listed, you can fill in the necessary parameters when you select the Add a Program menu option.

You must supply certain parameters, like the file name of the start-up program (with file extension EXE, COM, or—with Release 1.1—BAT) and the minimum amount of memory the program requires. For some of the other parameters, TopView has been programmed with default values (including not allowing the program to run while in the background but allowing the program to be swapped to disk).

The information can get a little technical at times. You need to supply the *maximum* memory an application can use, the window size, the number of screen pages used by the program, and the range of software interrupt vectors swapped. Whether the application writes to the screen directly rather than using DOS to handle the video interface must also be included. In some cases, filling in the parameters may require some educated (or uneducated) guesses and experimenting to see what works.

The program works tolerably well without a mouse but also supports one. TopView comes with some desktop utilities like a calculator and an alarm clock. The program does not show the image of a clock but does provide several alarms you can set with different "wake-up" messages with each.

TopView also has a DOS Services window with basic operations already included for files (copy, type, rename, erase, print) and directories (sort by name, extension, size, or date/time). TopView also provides an OTHER option, in which you can insert other frequently used commands. Certain DOS commands are not supported in OTHER. CHKDSK is one, presumably because of the dangerous results if the Fix option is turned on. The OTHER options you enter are saved for both current and future sessions. The DOS Services window doesn't simplify DOS all that much, however. This capability isn't one of TopView's strong points.

You can identify for TopView a group of programs that can be loaded when you select the group name. Version 1.1 can be

configured to load the group of programs when TopView is first started (like an AUTOEXEC.BAT).

IBM would certainly like TopView to become the standard operating environment. The program has a way to go yet. It also needs more applications written to take advantage of its features. But you have to remember that you are not just buying a hard disk manager when you invest in a product like TopView. You are also hazarding a glance at the future although whether TopView itself is *the* standard of the future is not clear at this point.

DESQview

One of the first things you may notice about DESQview from Quarterdeck Office Systems is the manual; its splashy modern cover invites you to open the book. Inside, the first few pages are written in a friendly style that suggests that this program is going to be fun. The instructions are straightforward, the examples illuminating, and the figures very clear.

Even the installation process has been made as simple and friendly as possible. You can choose a simple setup procedure that asks you only two questions: What kind of monitor will you be using? What brand of mouse (if any) will you be using? The installation program configures the program appropriately with defaults for every other setting. This feature lets you get up and running quickly. You can always fine-tune the other parameters later. (For example, the ratio of time allocated to background versus foreground tasks is changeable.)

One unique aspect of DESQview's installation program is that it automatically searches your hard disk for any applications DESQview already "knows" something about. The installation program notes the path to the program files and installs these programs as part of the menu structure. The program files are ready for you to execute the first time you start DESQview. You may need to double-check this process, however, in case DESQview mistakenly makes a connection between a file on your disk and one of its stock applications.

You can, of course, manually install a program too. You will have to specify some of the program's operating parameters to DESQview's "Add a program" module. Similar to TopView, DESQview stores the information about a program in a DESQview Information File (DVP). If your application program happens to include a TopView Program

Information File (PIF), DESQview can read and use the PIF file instead.

The installation process creates its own DV directory and puts all the necessary files there except for a lone DV.BAT file, which goes in the root directory. Of course, if you have a directory reserved for batch files, you can stick the DV.BAT file there. It creates minimal disruption of your own hard disk structure.

The second chapter in the manual, an 18-page quick tutorial, introduces you to DESQview's basic capabilities. The tutorial is a great way of getting started quickly. Although DESQview accommodates running with a mouse, you won't feel handicapped without one. Unlike using TopView, you don't need to move a menu bar to select a command or program name. Everything is selectable by a letter or two typed on the keyboard.

Figure 14.7 shows DESQview's initial menu, called up by a key or the mouse button. When either DESQview or you install a program, it is also given a one- or two-letter mnemonic to represent the program in the program menu (see fig. 14.8).

DESQview "does windows," so you can view multiple program sessions simultaneously, including programs in graphics windows. You can jump from one program to another by the press of a few keys. Windows can be resized, moved, shrunk, zoomed, hidden, and "put aside" (more on this last later).

DESQview also supports concurrency, in which more than one application can be *running* at the same time. Applications running

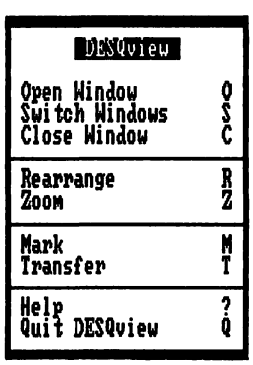


Fig. 14.7. *DESQview's initial menu.*

```
Open Window

BASICA              BA
DOS (128K)          D1
DOS Services        DS
GEM Desktop         GE
Lotus 1-2-3 1A      LT
Norton Util 3       NU
Sample Document     SD
Sample Sprdsheet    SS
SideKick            SK
Symphony 1.0        S0
WordStar            WS

Add a Program       AP
Delete a Program    DP
Change a Program    CP
```

Fig. 14.8. *DESQview's program menu.*

under DESQview need not be especially written to accommodate these features. They can be grabbed literally "off the shelf" and run unmodified. The documentation makes a strong point about this feature, perhaps to further distinguish DESQview from its competitors.

As a general rule, a windowing environment prefers well-behaved (with regard to DOS) programs. The environment tends to tolerate misbehaving programs by running them only in full-screen windows, suspending their operation when in the background, and managing cut-and-paste operations in a balky manner, if at all. Well-behaved programs can run inside a window and concurrently, whether in the background or not. DESQview has special *load* files that "discipline" selected misbehaving programs (WordStar and 1-2-3, for example). Any misbehaving programs for which you have no load file have to run in a full-screen window.

Concurrency is definitely a joy to watch. Admittedly, the concurrent operation is also slow to watch on a PC. But you can have both 1-2-3 and WordStar running side by side (or rather top to bottom, see fig. 14.9) on your machine.

The cut-and-paste operation can work between inherently incompatible applications programs. An advantage to DESQview's cut-and-paste operation is the capability not only to move the information but also to do some editing, filtering, and reformatting in the process. This capability is significant because the information can be transferred in a *usable* form. It is one thing to bring information into a spreadsheet, for example, and it is an entirely different matter

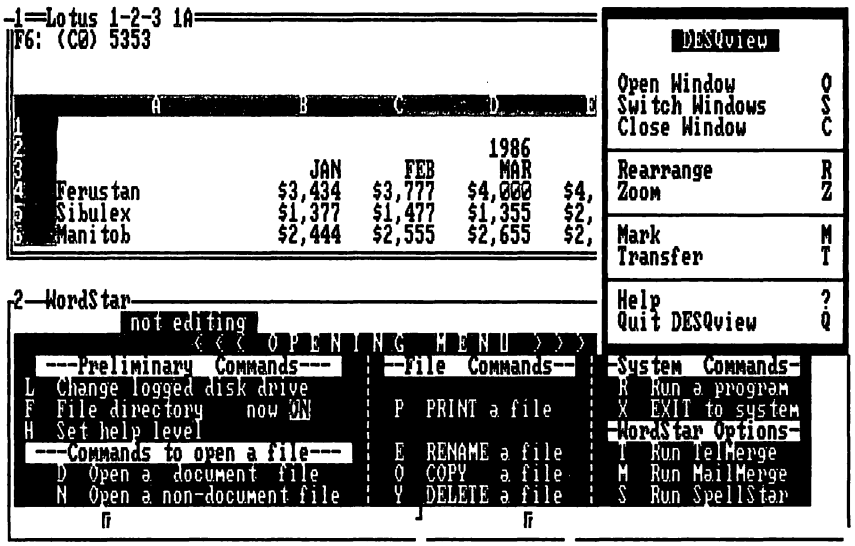


Fig. 14.9. *DESQview windows with 1-2-3 and WordStar.*

to import the file so that the data is split into the appropriate columns for later manipulation.

The documentation uses the cut-and-paste operation to demonstrate the Script option. This feature (usually called a macro capability) allows you to store a set of keystrokes that you perform to accomplish some task. When you need to repeat the task, you simply press a predefined key and DESQview does the task for you. Once you define the set of operations needed to transfer, filter, and reformat a record of data from a database program to a spreadsheet, for example, you can have DESQview perform the same operations for you on every other record.

If you want to run several programs that together consume more RAM than your system has, DESQview automatically swaps one program out to disk (often referred to as virtual memory). You don't have to close files and exit from the application. Of course, the application cannot continue to run when swapped to disk but is frozen at the point where the program is executing. When the application is swapped back, execution continues right where it left off.

If you temporarily don't need an application in RAM, you can choose to swap out an application. Virtual memory can work with extended and expanded memory (above the natural 640K DOS limitation) and a RAM disk. As mentioned in the discussion of RAM

disks, robbing Peter (stealing from the available RAM for running programs) to pay Paul (using a RAM disk for the program swap) makes little sense. Don't bother using a RAM disk for swapping applications unless you can place it in expanded or extended memory.

A DOS Services window in DESQview is already set up to handle file and disk management chores. After sorting and displaying a particular directory, you can tag or mark files on the screen to be operated on. Along with standard BACKUP, COPY, ERASE, PRINT, RENAME, and TYPE commands is an APPEND command, which is used to join two or more files.

The program has a DOS submenu containing some directory commands (CHANGE, MAKE, and REMOVE), as well as commands to copy a diskette, format a diskette, and restore backup files. When you mark files for tasks, you can do so individually or by specifying a filespec. Then you can instruct DESQview to display only the marked files for convenient viewing.

Initially, problems may arise getting the DOS Services window to work properly. Among other things, the characters on the screen may refuse to stay confined to the window, and they may spill out of the window borders. One reason might be that your display adapter card is equipped with an early version of the display software. That card won't accommodate DESQview's screen management operations. (How could it if it predates DESQview by months?) This example makes the point that problems you have with an environment package may be caused by things other than the package itself. If you install the updated version of your display adapter software, everything will work smoothly.

DESQview is a pleasure to learn and a pleasure to use. If you are considering a windowing environment with multi-tasking, this product deserves serious consideration.

Summary

This chapter reviews some popular programs that are categorized as enhanced operating environments. The significant points presented are

- The primary goal of these enhanced environments is to act as a "shell" around DOS and to make easier and more efficient your interaction with DOS.

- The enhanced environments offer hard disk management tools and different combinations of more complex features: windowing, switching between applications programs, transferring data between programs (cutting and pasting), and multi-tasking.
- The major software entries in this field are IBM's TopView, Microsoft's Windows, Quarterdeck Office Systems' DESQview, and Digital Research's GEM Desktop.
- GEM and Windows require some kind of graphics display; TopView and DESQview will run on virtually any display device.
- GEM and Windows can use icons as helpful visual cues; DESQview and TopView are strictly menu-based.
- GEM is a Macintosh-looking environment, complete with icons and optional though almost essential mouse. With GEM acting as a simple front-end menuing system for DOS, you can manage your files on a hard disk and call up applications programs from within the GEM desktop.
- Windows also offers a Macintosh-like environment and includes hard disk management tools, windowing functions, the ability to call up applications programs, and a certain measure of multi-tasking.
- TopView is a text-based system that includes hard disk management functions, windowing, the ability to call up applications programs, and multi-tasking. It has the provision for cutting and pasting between programs but it works with only a limited number of programs at present.
- DESQview is a text-based system that appears to have been developed as a direct alternative to TopView. It has hard disk management tools, windowing, the ability to call up applications programs, multi-tasking, and a workable cut-and-paste facility.

Chapter 15

Backup

Throughout this book I have climbed on the operational soapbox and given you reasons for backing up your hard disk. Backup is like insurance: you pay the price because you think something bad can happen. You hope this prophecy is not self-fulfilling; but between normal human error and electromechanical breakdown or accident, something bad certainly can happen. The question is not *whether* to back up, but *how*, *what*, and *when* to back up.

This chapter examines the how, what, and when. The text describes some of the different ways of backing up files and the tradeoffs for these methods. The "best" method for you (the how) hinges on

1. The number of files you need to back up routinely
2. The sizes of the files to be backed up
3. Your particular machine configuration
4. The extent of convenience you demand
5. The amount of money you wish to spend

The what and when depend on these factors and also on how much effort is needed to replace lost files.

If the number of files is small and no file is larger than the size of your backup media, a simple file-copying procedure works well. The

simple copying procedure becomes burdensome when too many files must be copied, the files span too many directories, or a file is too large to fit on a single diskette.

Obviously, 1.2M diskettes can hold more and larger files than 720K diskettes, which hold more and larger files than 360K diskettes. If the number of files is very large and your diskette media small, you will want to consider software and hardware alternatives: fast backup programs, removable-media hard disk drives, and cartridge tape drives. Some alternatives offer greater convenience for any system; however, the prices of these alternatives are frequently high. Unfortunately, price does not necessarily dictate the level of convenience. Some inexpensive solutions can be very convenient.

Whatever decision you make, remember that the backup process has two parts: backup and restoration. The second process is impossible without the first, and the second process is the one that actually "saves the bacon."

No matter what mechanical or software system you use, spend some time analyzing both the backup and the restore processes. Make sure that you understand the equipment and the software. Be sure that the system does what you think it should do.

Explore your system by simulating the backup process for a few days and experimenting with the restoration process. Experiment while the tolerance for errors is large; the tolerance level narrows greatly when the backup and restoration is real.

Establishing Guidelines

No matter what method you use for backup, you should establish operational guidelines. The method of backup (the how) affects the what (files) and when (how often). You start by analyzing your needs based on your file use. What files need to be backed up? Do some files demand more frequent backup? Where should the backup copies be stored?

After you pass the stage of routinely copying a handful of files, backing up all the files on your hard disk is burdensome. Selective backup becomes appealing. Large data processing installations do not back up their entire disk storage each time. Why should you? Obviously, unchanged files (such as DOS utilities or applications program files) and data files that are updated rarely can be backed up infrequently. Files that are changed daily or weekly should be backed up frequently.

The best plan is to adopt the same basic strategy that large data processing installations use. They run two different types of backups. The first type is a total backup, a *master backup* of all files on the disk. The second is an *incremental backup* of the files that have been changed or created since the last total backup.

The two procedures are easy to implement. First, you back up the entire disk. This backup copy becomes the *master backup*. The total backup takes the most time but is done infrequently, say every month or so. You make the more frequent but less time-consuming incremental backups as required. Therefore, the regular backup process takes less time.

The real payoff comes when you must restore the files on your hard disk. You first restore the master backup set, then the incremental backups. Depending on your backup scheme, only two (or a few more) backup sets restore all your hard disk files.

How do you select the files for the incremental backup? You select the files that have been created or changed since the last backup. Must the creation or modification date of every file be compared to the date of the last complete backup? Yes, the comparison must take place. Must you manually make this comparison? Not at all.

Whether the program handles diskettes, tape cartridges, removable hard disks, or video tapes, virtually every backup and extended-copy program (except DOS's COPY) offers the option of processing only changed or new files. DOS provides two different ways to identify these files. The first way is by the DOS archive attribute, which is set when a file is created or changed. The second way is by the file's creation or alteration date and time, which are stored in the file's directory entry.

The final step in planning for incremental backups is classifying files by their frequency of change and their value and then determining how often each group should be backed up. You can eliminate rarely changed files from incremental backup; these files are caught by the master backup.

Files that are changed daily and are difficult to re-create should be backed up frequently—daily, every few days, or weekly. You weigh the premium of the insurance (the time spent in backup) against the benefits (amount of time saved by restoring the file compared to re-creating the file). If the data cannot be re-created, the benefits far outweigh the premiums; and you should back up this group daily or every few days. If the file is easy to re-create, you can back it up weekly or even monthly.

A trick that speeds incremental backups of "high value" files is to isolate these files by application in separate subdirectories. For example, place your most valuable 1-2-3 files in a directory separate from other worksheets. Place your most valuable letters or contracts in a directory separate from other documents. You can select only files from these "high value" subdirectories to back up every day or two. Your weekly or monthly backup catches all new or changed files.

The two-part process—complete backup combined with incremental backups—takes much of the pain out of getting into the habit of regular backups. You can establish a routine that consists of a monthly total backup and daily incremental backups (or every couple of days, depending on how often you use your computer).

The next question to consider is where to keep your backups. If you use two master sets of backups, keep one set near the computer and one set off-site. The set near the computer can be placed in a desk drawer, a file cabinet, in a floppy diskette holder in the corner of the room, or wherever is convenient. This on-site set should be readily available should any problems arise. The most recent master backup set should be kept on-site. The off-site set, which is usually the less recent backup set, can be stored at home or in any other safe location, including a bank safety deposit box. Generally, keeping the second master set at home is sufficient security.

You keep one backup set off-site in case of a disaster at work. For example, one author had a fire at his workplace. Although the fire did not reach the room holding the equipment and diskettes, smoke from the fire did. The smoke, of course, penetrated locked file cabinets and made many diskettes unusable. Disasters are rare, but their effects are devastating. A small amount of paranoia can be healthy.

Ensuring Backup Integrity

You should make backup copies as if your hard disk will be stolen overnight. The "thief" might be an accidental operation, an electronic malfunction, or a literal theft of the computer. You must have absolute faith that the backups are accurate representations of your files. The backup copies must be error free. You cannot assume that a mistake can be corrected later. The opportunity to correct mistakes disappears when the hard disk drive "disappears." When the hard disk is down, the only thrill of discovery is that the hard disk files can be restored quickly and without errors. Any other discoveries are traumatic.

Electromechanical devices (disk or tape drives) can make faulty recordings. All devices offer some form of dynamic verification. As discussed in Chapter 6 about COPY, these verify-while-recording schemes are not 100 percent reliable, but they are close. Another scheme is to record the information redundantly, placing from 4 to 16 copies of the data on the media. Even if the media becomes badly flawed, at least one copy will be accurate. You also can use extensive error-checking and correcting schemes so that the program can reconstruct accurately the information from even badly damaged media.

If you use a DOS utility or a similar program to back up files, consider using the verify feature. DOS does not incorporate sophisticated error-recovery schemes in its records. The verify option provides a better chance that the backup copy is good. If you use a command like COPY for backup, you can include the */V* switch to turn on the verify option. If a */V* switch is unavailable, you can use the DOS VERIFY ON command to enable verification while the COPY command works and then issue VERIFY OFF. This command sequence is described in Chapter 6 in the discussion of the COPY command.

As usual, you have the tradeoff of accuracy for speed. VERIFY ON doubles the time COPY needs for the backup operation. You have no excuse, however, for sacrificing data integrity during a backup operation.

Some people take the middle-ground position: invoking VERIFY ON or using the */V* switch during monthly backup and using VERIFY OFF (or omitting the */V* switch) during daily backups. Others go the route of issuing VERIFY ON for backing up data files but using VERIFY OFF for program files. The philosophy is that the backing up of program files is strictly a convenience because these files remain intact on the original program diskettes.

The question of verification is one you must determine for yourself. If the files are important enough, you probably won't want to risk adulterating the backup process with some unseen disk-write problems. If your disk drives have proved themselves reliable, however, the extra time spent for verification may not be worth the trouble. In any case, remember that you'll normally want to turn off verification when you are finished backing up.

Verification affects only disk drives. DOS has no such control over other devices. Disk cachers also may defeat the purpose of verifying or comparing disk-based files. Because a disk cacher holds strategic parts of the file in memory, when DOS verifies or compares files, it

may be comparing backup copy with the in-memory copy of the file and not with the disk-based version. If you feel uncomfortable about the accuracy of verification, disconnect or temporarily disable the disk cacher.

Summing Up Backup Guidelines

Summed up succinctly, the guidelines are that you perform the backup process so that the restore process is possible. You restore files because the working copy of the file is damaged or missing. The backup process is analogous to paying insurance premiums; the restore process is the insurance compensation. The object of the process is to make the backups as accurate and up-to-date as possible but invest the minimum amount of time.

The best method is to make infrequent master backups of your complete hard disk and more frequent incremental backups of files that are changed. The master backups capture your entire hard disk; the incremental backups capture the files that have been changed since the master backup.

The incremental backup is performed as often as necessary, based on the re-creation time and the cost of the files involved. If files can be easily re-created, the incremental backup can be performed weekly or every few weeks. If the files cannot be easily re-created, incremental backups should be performed nightly, every few nights, or weekly.

To vary the incremental backup process, you can back up most frequently only the high-use files (files that vary most often), and back up less frequently files that are changed only every couple of days. This method requires the least amount of time spent in performing incremental backups.

The master and incremental backups should be stored in a convenient but secure location. If two master backups are maintained, the less recent backup should be stored off-site. All sets should be safe from mishandling, confusion (mixing of diskettes from different backup sets or diskettes that do not hold backups), and damaging influences (such as heavy electric motors).

Because the integrity of backups is critical, you must take every precaution to ensure that the backups are correct. Although DOS's VERIFY is not one hundred percent accurate and slows the backup process, you should use VERIFY if no other form of verification is possible.

The following sections discuss specific backup programs. Depending on the number or size of the files to be backed up, the venerable DOS COPY or the BACKUP, RESTORE, and XCOPY programs should be used. COPY is discussed first, BACKUP-RESTORE second, and XCOPY last.

Using DOS COPY for Backups

Suppose that after analyzing your needs, you find that most of your storage is devoted to program files and a few data files. An easy and effective method for you may be to use the DOS COPY command and ordinary 360K diskettes. After each work session, you back up only your data files. This method is quick, easy, and cheap, and it serves the purpose.

Although this procedure sounds too easy, it may be very realistic for your purposes. The original program files are on the floppy diskettes in the applications package. You can reinstall these files the way you installed them the first time. If, however, the thought of re-creating all those subdirectories, recopying the program files, reinstalling the programs, and then copying your data files seems intimidating, you should consider other alternatives. Re-creating the contents of your hard disk may be more work than you first think.

Because COPY accepts wild-card characters (? and *), the command is flexible. Using COPY to back up files in one or two directories is convenient. Restoring files with COPY is a simple reversing of the procedure; you just copy the files from the backup diskette. If you change only a few files a day, you can establish an effective backup procedure by copying the changed files to a floppy diskette at the end of each day.

This use of COPY means that you select files by name or by directory. You forgo DOS's ability to select recently created and changed files. If your changes are isolated to one or two directories, COPY works well.

When you use COPY, remember that DOS occasionally has amnesia. To ensure that the recording is correct, use the DOS verify functions or the COMP command. As previously discussed, you can use the */V* switch or issue a VERIFY ON command. For example, you can use one of these commands to invoke DOS verification:

COPY C:\WORDS*.* A: /V

or

```
VERIFY ON
COPY C:\WORDS\*.* A:
VERIFY OFF
```

The two sets of commands have identical effects.

The most thorough method is to use the COMP command to perform a byte-by-byte comparison of the copies with the original files. The command sequence is

```
COPY C:\WORDS\*.* A:
COMP C:\WORDS\*.* A:
```

If you feel that COPY fits the bill, you may wish to construct an end-of-day batch file to automate the process. This batch file has the commands needed to copy the files that have changed. If you use this procedure, remember to update the file when you add new programs or new subdirectories to the system. The batch file must reflect any new backup needs. Remember also the warnings about disk cachers and the way COPY V2.1 mishandles every 256th file.

Using a Better COPY Command

Under the general classification of using COPY to back up files come several commercial, shareware, and public-domain programs that "copy" files. For example, PCOPY can be used to select the files to copy to backup diskettes (see Chapter 10). G-WHIZ and other hard disk managers that use menus also can be used for selective copying (see Chapter 13). If your situation warrants, these programs can replace numerous repetitions of the COPY command.

BAC.COM, a clone of COPY, provides some additional features for a "poor man's" incremental backup. The BAC syntax is similar to the COPY command but improves on COPY in two key ways:

1. BAC prevents copying an older version of a file over a newer version. For each file that might be copied, BAC determines whether the file already exists on the target disk. If the file exists, BAC copies that file only if the target file's date is earlier than the source's.
2. If the target disk fills up, BAC prompts you to insert another diskette so that BAC can complete the copying process. COPY simply aborts when the target diskette is filled.

BAC is inappropriate for a master hard disk backup. This program is no speedier than DOS's COPY and cannot handle files larger than a diskette. As with COPY, however, BAC files do not need to be restored by a special program (COPY works well). BAC also has more features than COPY.

BAC.COM is described in *PC Magazine*, Volume 4, Number 17, and is also available from the magazine's bulletin board.

While BAC can be called the poor man's solution to using COPY for backup, other solutions exist. The next solution is the "granddaddy": DOS's BACKUP and RESTORE combination. The following sections discuss these two commands and the extended XCOPY program. Other programs for backup are described in Chapter 16.

Using DOS BACKUP and RESTORE

The BACKUP and RESTORE programs are a pair of powerful utilities that are bad-mouthed almost as often as the downtrodden EDLIN, the line-oriented text editor also provided on the DOS system diskette. The most frequent complaint is that BACKUP is slow. Less frequent complaints refer to various flaws in the different versions of pre-V3.2 BACKUP. BACKUP V2.1 and V3.0 had problems with some resident programs. BACKUP V3.0 and V3.1 occasionally botched an important file or the backup files themselves. However, IBM has issued corrections for these problems.

Regardless of earlier problems, BACKUP has capabilities COPY cannot mimic. BACKUP can

- Back up files larger than a diskette
- Provide better file selection
- Traverse the subdirectory tree

A minor drawback to BACKUP is that the backup files are not directly usable. You must run RESTORE to re-create the files intact. This drawback is the "nature of the beast" and is really no problem at all.

Although BACKUP can back up to another hard disk or back up 1.2M or 720M diskettes to another type of diskette, the principal use of BACKUP is backing up a hard disk to floppy diskettes. A not-so-trivial drawback is that BACKUP requires formatted diskettes. BACKUP cannot gracefully resume operation if you must stop the program to format a diskette.

You calculate the approximate number of diskettes needed by totaling the size of the files you'll back up and dividing this figure by the capacity of the diskettes you will use. If you plan to back up most or all of your hard disk, use the figure CHKDSK supplies for total bytes used by "user" files. For selective backups, use the DIR or an extended directory command and add the bytes for the files you want.

Take the total number of bytes to back up and round that number to the next higher K (1,000 bytes). For example, 835,984 bytes becomes 836K. Divide this rounded byte count by the capacity of the floppy disk drive—360, 720, or 1200. The answer, rounded to the next higher whole number, is the number of diskettes to prepare. For example, 836 divided by 360 (the figure for a double-sided disk drive) is 2.3, or 3 diskettes.

Because you are approximating, compensate for errors on closer answers (a fraction above 0.5) by adding one diskette to the answer. In fact, if you have any doubt in your calculations, add an extra diskette.

If you are backing up a full disk drive, consult table 15.1 for a list of the maximum number of diskettes needed for the backup.

Table 15.1
Number of Diskettes Needed for Backup

Diskette Capacity:	*360K*	*720K*	*1.2M*
10M hard disk	29	15	9
20M hard disk	59	29	18
30M hard disk	83	44	27
40M hard disk	116	58	35

The BACKUP Command Syntax

The complete BACKUP command syntax is

*dc:pathc***BACKUP** *ds:path**filename.ext* **dd:** */A /M /S /D:date*

The *ds:path**filename.ext* is the name of the file to back up (the source). At least one element of the full file name must be given. If you omit the disk drive name, the current disk drive is used. If you omit the path, the backup starts with the current directory. If you omit the file name, *.* is assumed.

The **dd:** is the destination for the backup files. In DOS V3.0 and later, **dd:** can be any disk drive; in earlier versions only floppy disk drives can be specified. The switches for BACKUP are listed and defined in table 15.2.

Table 15.2
BACKUP Switches

Switch	*Function*
/A	*Append* the backed up file(s) to the destination. Without */A* all files on the destination are erased before new backup files are added.
/M	Back up *modified* files only (files with the archive attribute set)
/S	Traverse the *subdirectories* (used to back up all files in the subdirectory tree)
/D:date	Back up files created or changed since *date*. The form of *date* depends on your setting of the COUNTRY option in the CONFIG.SYS file and may be *mm-dd-yy* North American *dd-mm-yy* European *yy-mm-dd* Far East where *dd* is the day, *mm* the month, and *yy* the year. The slash (/) may be used instead of the minus sign (-).

The switches give you better control over which files will be backed up. For example, the following command causes all files in only the root directory of drive C to be backed up to the diskette in drive A.

BACKUP C:\ A:

With all DOS commands, DOS limits its action to files it can "see" (files in the current directory). To expand DOS's vision to traverse the subdirectory directory, you must add the */S* option. To back up all files on C, the command is

BACKUP C:\ A: /S

DOS backs up the root directory and then traverses each subdirectory and backs up its files.

The */A* switch affects the treatment of the destination. If the destination is another hard disk, the subdirectory \BACKUP houses the backup files. Giving the */A* switch preserves the files already in \BACKUP; omitting the */A* switch causes the program to destroy the files in \BACKUP. If the destination is a diskette, all files on the diskette are erased.

The */M* and */D:date* switches control which files of the subdirectories are processed. These switches are discussed later in this chapter.

BACKUP adds to the beginning of each file a header telling the file's original subdirectory. When restoring files, RESTORE reads each file's header and places the file in the correct subdirectory. The header also holds information about the file's size and the number of diskettes it spans, in case the file must be split. The file cannot be copied or reused unless you remove the header.

When you are using the BACKUP command, designate the starting directory with care. BACKUP, like many other traversing commands, starts with the current or designated directory and works down the subdirectory levels. If you use a lower subdirectory (forget to specify a directory or specify a different starting directory), you get only part of the files on the disk.

Often you may want to use a different starting directory to back up only strategic parts of the disk. If you want to process the whole disk, however, you must start at the root directory.

Recommended BACKUP and RESTORE Mode of Operation

The options of BACKUP and RESTORE (RESTORE is discussed in a following section) offer a variety of procedures. Regardless of which method you choose, I strongly recommend that you keep several sets of backups. You should have at least one set (possibly two) of media for your master backup. Depending on your mode of incremental backup, you may need as many as five sets (one set for each working day of the week). Generally, two sets are sufficient. The reasoning behind multiple backup sets is that if one set is damaged or if the recording is bad, you can fall back on the additional set(s). Redundancy is the key to safety.

Remember that diskettes must be formatted before they are used with BACKUP. Forgetting this step is a trivial problem if you stop BACKUP just after it starts. But running out of formatted diskettes is

costly if you must stop BACKUP half way into the process and restart the program from the beginning.

With some versions of BACKUP, you may need to disconnect certain resident programs, such as Borland's SideKick®. Check the documentation that comes with your program. To my knowledge, BACKUP V3.2 works with almost all popular background programs.

I also suggest that you use a disk-caching program or have your CONFIG.SYS file set 20 or more file buffers. Disk cachers and file buffers speed the backup process. You also may need to "uninstall" copy-protected programs. Copy protection is discussed a little later in this chapter.

The Master Backup

How frequently you need to make a total master backup depends on how much your computer is used. Estimate how many files on your disk are changed each week. If, over a period of four weeks, fewer than 25 percent of your files are changed, you might perform a master backup every other month. If the figure is greater than 50 percent or you have added a major multifile applications package to your hard disk, make the master backup more frequently—monthly or bimonthly.

I suggest that you always make the master backup on the same day or days of the month. If you use the date option later, you can more easily remember the date to specify to BACKUP.

Be sure that you have enough formatted diskettes. Better to have too many than too few. Then label the diskettes with numbers from 1 to the end of the set. This numeric labeling can help in the restoration process. As previously explained, I highly recommend that you invoke VERIFY ON before you start the backup procedure.

The command you use is

BACKUP C:\ A: /S

This command traverses all subdirectories and copies all files from the hard disk. To copy your backup files on diskettes in drive A, place the diskette labeled "1" in the drive and use the diskettes in ascending order. Then put each full diskette in a separate box, date the boxes, and store them in a safe place.

After you have backed up, you might also run a TREE command on the hard disk to list which files have been backed up. To print a copy of the subdirectories and files, enter the command

TREE /F >PRN

Keep this printout with the diskettes.

If you use more than one master backup, rotate the sets. Use the first set for your first master backup. When you perform your next master backup, use the second set, and so forth. This method ensures that you have a set of backups in case something drastic happens to the other set.

The Incremental Backup

You also will need to decide when to make incremental backups. Critical files should be backed up daily or every couple of days. If your files are less critical, you may wait longer; but don't wait too long. If you ever get the nagging feeling you should back up your hard disk, you've waited too long.

You have several ways to control which files are backed up. One method is to choose the directory (or directories) to back up. Unfortunately, BACKUP allows only one full file name. If the files to back up are on different parts of the subdirectory tree, you must issue the command many times.

The other methods determine which files in the specified directories should be backed up. These methods use the DOS archive attribute and the file's creation or alteration date in the directory. With both methods, you can select the files that are new or have been changed since the last backup.

These methods of incremental backup are based on two switches. The */M* switch directs DOS to back up any files that have been modified or created since the last backup. BACKUP examines the files' archive attributes and selects the files that have the archive bit turned on. With the */D* switch, you can select files that are new or have been changed since a specified *date*.

The /M Switch

The first method uses the */M* switch. DOS grabs all files with the archive bit turned on. Before jumping to the conclusion that this switch consistently grabs all changed files, remember that BACKUP automatically turns off the archive bit when the command processes a file.

Suppose, for instance, that you make a total backup of your hard disk on May 1. On the evening of May 2, you use the */M* switch to pick up only files that have been modified since the day before. On May 3, you do the same. Do you have all changed files? Maybe not. If a file was changed on May 2 but not on May 3, the file is backed up

only on May 2. If you reuse the diskettes, BACKUP wipes out the previously backed up files before writing the new set. The files changed on May 2 (but not on May 3) are lost.

The use of the */A* (append) switch is therefore imperative. You must maintain the previously backed up files. The */A* switch forces BACKUP to keep the previously backed up files on the backup diskettes.

The appropriate command for the day after a master backup (or the first time the incremental backup set is made after a master backup) is

BACKUP C:\ A: /S /M

For subsequent days the command is

BACKUP C:\ A: /S /M /A

Both commands select only changed or new files. The second command adds the files to the backup set. You use the first command when first using this incremental backup set after a master backup. The master backup has all your files. You want only the changed files. When you use the first command, you begin by inserting the first diskette of the series. The out-of-date files are erased from the diskettes.

When making the second backup, you include the */A* switch to keep the established files. You insert the *last* diskette from your incremental backup set. BACKUP does not need the others. A temporary label on the last diskette or piece of paper inserted before it helps identify the diskette to use.

With this method, you should use only one set of diskettes for the incremental backup. The chances of out-of-date files replacing more recent files are too strong if you restore from two incremental backup sets.

No harm is done if you back up the files more than once. If a name conflicts, DOS changes the extension to @01 for the first conflicting name, @02 for the second, and so on. Because the different versions of the files are backed up sequentially, you always restore the latest version of the file. For example, suppose that FILE01 has been backed up six times. You'll end up restoring all six versions, but RESTORE overwrites a previous version with the more current version. Version one is replaced by version two, two by three, until version six replaces version five. The */A/M* scheme works.

The /D Switch

The */D* switch is used to select all files that have been created or changed since a given date. The */D* switch, in North American form, is */D:mm-dd-yy*, where the month, day, and year are specified. The date you use is the date of your last master backup or that date plus one day. BACKUP selects files with dates in the directory matching or later than the specified date. If you performed the backup as your last act of the day, no files were changed on that date, so you use the day after the master backup date. If you created or changed files on the day of the master backup, use the master backup date.

Assuming that your master backup was June 1, 1987, the command is

BACKUP C:\ A /S /D:6-1-87

This command selects all files changed or created on or since June 1, 1987. As you create a new master backup, you change the date accordingly.

If you use this command each day, you should have more than one incremental backup set. Use the first set on the first day of incremental backup; use the second on the second day, and so on, and rotate the sets. Because the */A* switch is not used, the backup diskettes are erased before files are added. Using incremental sets ensures that if one set is damaged, a previous set will contain a recent copy of the files.

Prelabel your diskettes in ascending numeric order, and use a sticker to show when the backup was made.

Advantages and Disadvantages of /M/A and */D*

Both */M/A* and */D* have advantages and disadvantages. The */M/A* method is the quicker of the two for backup. Only the files that have been changed are processed. */D* takes longer for backup. As the number of changed files grows, */D* processes more files on a daily basis.

/M/A is at a disadvantage when restoring files. RESTORE takes longer because each version is placed on the hard disk and then discarded if the program finds a later version. */D:date* is faster when you are restoring files. At most, two copies of the file are restored: the copy from the master backup and the copy from the incremental backup. This procedure is discussed in the section on RESTORE.

/M/A takes more diskettes than */D*. As more files are added to the backup, the number of diskettes used grows. */D* never uses more

than the number of diskettes needed to hold the entire contents of the hard disk. */M/A* can actually use several times this number.

When you are restoring files, */D* is also faster if the file has not been changed at all. At most, two sets of backup diskettes are searched: the last incremental set and the last master backup. The single search through the last */M/A* incremental backup can be longer than the two-set search.

If you unknowingly erase a file and do not notice for several days, */M/A* is more reliable for restoring the file. With */D* you erase the previous incremental backup set. If you repeat the erasure process on all incremental sets, the only version of the file left is on the master backup. With */M/A*, the set is never erased, and a more recent version of the erased file can be restored.

/D offers greater backup safety through redundancy. At least two incremental backup sets are used and two copies of the files are made at regular intervals. If the most recent backup is damaged or bad, the next most recent backup set can be used. If the single */M/A* set is damaged, some files on the incremental backup are risked. However, both methods will be plagued if the master backup becomes damaged.

Problems with BACKUP

No matter how you choose to do your backup, you may run into some problems, depending on the version of pre-V3.2 PC DOS or MS-DOS you are running.

Early copies of BACKUP V3.0 experience memory lapses every so often. This BACKUP blew its required heading file, called BACKUPID.@@@. If you are using BACKUP V3.0, do a directory of each diskette. BACKUPID.@@@ should be the first file and should have a length of 128 bytes. A discrepancy means that the backup set is bad, and BACKUP must be run again.

BACKUP V3.1 had different "flakes." BACKUP would report an "invalid disk specification" even though you specified the command correctly. Conflicts with some early versions of resident programs like SideKick were the cause. Removing the program before running BACKUP eliminates the problem.

BACKUP V3.1 had another quirk with read-only files. If BACKUP had to split a read-only file over two diskettes, BACKUP botched the job and that file could not be restored. If you use read-only files, look for that file name as the last in the list of one diskette and as the second file (after BACKUPID.@@@) on the next diskette. If the

combined size of both is not equal to the original size plus 256 bytes (the two headers, 128 bytes for each of the two files), BACKUP messed up again.

IBM has issued revamped versions of BACKUP for DOS V3.0 and V3.1. Consult your local dealer for these programs.

Some MS-DOS versions of RESTORE also have problems. For example, the conflicts between certain versions of the ROM BIOS for the Sperry IT (a Personal Computer AT clone) cause RESTORE V3.1 to fail when files span more than one diskette. The backup is intact, but the files cannot be restored.

A Special Note about Copy-Protected Programs

Although copy protection is on the wane, some software manufacturers still feel that such protection is necessary to prevent pirating by unscrupulous individuals. The perceived need for copy protection, however, clashes with the convenience of the hard disk and its ease of operation.

The "old fashioned" method of copy protection required the original program diskette in drive A to start the program as proof you were using a legitimate copy of the program. This *key disk* requirement satisfied software manufacturers but leaves hard disk users unsatisfied. Needing a key diskette in the disk drive negates part of the hard disk drive's convenience.

To accommodate hard disk owners, software firms devised a different copy-protection method. The method allows the program to be installed and run from the hard disk. These installation programs have a fixed number of times you may install the software, some only once or twice. Before completing the process, the installation software writes on a hidden area on the diskette the number of installations left. The intent is to prevent the software from being installed on more than one computer.

These protection schemes depend on hidden files placed in the root directory or on the exact hard disk sectors where key files are placed. When schemes depend on pieces of program at specific spots on the disk, you cannot back up and restore these programs. The hidden files or the master files are almost never restored to the identical sectors of the hard disk, and the chance of successful restoration is less than one in several million. Therefore, backing up and restoring these copy-protected programs is folly.

The answer is either don't restore copy-protected programs or don't back them up. You cannot use RESTORE on the copy-protected program files, but you can use RESTORE on all other files. You can back up all the program files and then use RESTORE's */P* switch (discussed in the next section) to restore all but the copy-protected files because most of the copy-protected files are marked read-only. **RESTORE /P** will query you before restoring any read-only file. If the file's name is bizarre (such as VDF0300.VDF, CML0210.HCL, or SGS0230.SUP) or does not normally appear in the directory (such as 123.EXE where 123.COM does show in the directory), don't restore the file.

Your other choice is not to back up the files at all. You either skip the copy-protected files when backing up or first "uninstall" the program and then back up the disk. Fortunately, installable copy-protected programs provide a program to reverse the process, "uninstalling" the application when needed, so the count is not run down to zero. For example, 1-2-3 can be installed on the hard disk, once from the provided system disk and once from the provided system backup. 1-2-3 can also be uninstalled back to the system and system backup diskettes.

After you uninstall the copy-protected program, you may back up the hard disk without problem. The uninstall process removes the protected files, so the backup program does not process files that cannot be restored. You must then reinstall the program after the backup.

The safety play is to uninstall copy-protected programs before you make the master backup, and do not process copy-protected files during an incremental backup. Because these files are not changed, you need not copy the files during the incremental backup; and backing up files that cannot be restored makes little sense anyway. Also follow this procedure when cautioned by a program publisher, such as when using certain disk-optimizing programs, or when you deliberately reformat the disk (covered in the section on disk optimizers in Chapter 12).

The uninstall procedure (or not restoring copy-protected files) works well when everything else works well. The fault with either procedure shows up when the hard disk finally "blows a gasket" and fails with a copy-protected program still on the disk. Your only choice is to install another version of the program, reducing your installable count. If the trouble with your hard disk is repeated, you will exhaust the number of allowed installations. This quandary faces anyone who uses copy-protected software on a marginal hard disk.

If you know failure is coming, you can uninstall the program. If the program can be invoked from a floppy diskette, you can continue operations. If the program runs only from a hard disk, however, you gamble because the program must be kept on the hard disk to continue operations. You leave the program on and hope that sufficient notice is given by your hard disk to uninstall the program before rigor mortis sets in on your hard disk.

Any operation that disrupts the copy-protected program reduces the magic count by one. If the copy-protected files are "trashed," you must reinstall the program. When the count drops to zero, no matter what the cause, operations stop. Fortunately, most software publishers provide additional program disks upon presentation of proof of purchase and a small fee.

The RESTORE Command Syntax

Ideally, you'll never need to restore a file. However, such dreams are empty. Someday, somehow, you'll need those backups. The advantage to the BACKUP and RESTORE programs is that you do not need to remember where all your files came from. RESTORE reads the heading at the start of each file and restores the file in the subdirectory it came from. RESTORE automatically creates that subdirectory if necessary.

The syntax for the RESTORE command is similar to, but simpler than, BACKUP's:

*dc:pathc***RESTORE ds:** *dd:path**filename.ext /S /P*

The **ds:** is the name of the disk drive holding your backup files. The *dd:path**filename.ext* is the name of the file you wish to restore. You can omit the disk drive name if the hard disk is the current disk drive. The *path*, if given, specifies where to start the restoration process. If a file name is not given, *.* is assumed.

The two switches for RESTORE are

/S Traverse the *subdirectory* tree to restore files (identical to BACKUP's */S*)

/P *Pause* and ask (query) whether a file should be restored if it has changed since the last backup (the archive bit is on) or the file is marked read-only

If you are restoring a large group of files (the entire disk or an entire subdirectory tree), use the */S* switch to traverse the

subdirectories. If you wish to restore the files in a named or current subdirectory, omit the */S* switch.

The */P* switch causes DOS to ask your permission before restoring read-only files or files with the archive attribute turned on. Most read-only files, including copy-protected files, are marked so that the file is not changed. The chances are strong that files marked read-only are current and need not be replaced. In the prompt, RESTORE states that the file is marked read-only or that the file on the hard disk is more current than the backup copy. Copy-protected files, if backed up, should never be restored.

The archive bit query helps prevent overwriting a more current file with an out-of-date backup file. For example, suppose that some files in a subdirectory are erased. Rather than restore all files without restriction, you can save more recent files from being replaced by using the */P* switch. DOS queries whether the file with the read-only or archive bit on should be replaced by the file on the backup diskette. The program waits for you to enter a **Y** to replace the file or an **N** to skip the file. Unless you have completely lost that particular file (an out-of-date version is better than no version at all), you will probably want to enter **N**.

The Restoration Procedure

The actual restoration procedure starts with identifying the backup set or sets to use and the method for your incremental backup. The number of sets needed depends on the breadth of the restoration (number of files to restore) and the number of backup sets holding the most current copies. If you know that one specific set holds all the files to be restored, you use just that set. Most frequently, the last incremental set is used. If you have a printout listing the files on each diskette, the task of identifying which set holds which files is easier.

If you are not sure which set holds the files or whether the files are on two different sets, use the oldest set first and then the most recent set. For example, you can use the date approach (*/D*) with BACKUP. You first restore the files from your most recent master backup. Then you restore the files from your most recent incremental backup. If you use more than one incremental set, the last set is the most up-to-date. All files that have been created or changed since the last master backup are on the latest incremental set.

If you use the archive-bit approach and need files from your master backup, you restore first from your master backup and then from the incremental backup set. As previously explained, no problem arises if the file exists several times on the incremental backup files. Although all versions of the file are restored, only the most recent version remains after the restoration is finished.

Whatever approach you use, RESTORE always wants to "riffle" (examine) each diskette in the backup set. Unless you know exactly which diskette holds the start of the files (rarely true), you must start with the first diskette. RESTORE prompts you to insert the diskettes in order. If you get the diskettes out of order, RESTORE informs you and asks you to place the proper diskette in the disk drive. Numbering your backup diskettes helps you avoid this problem.

The file name you give to RESTORE is the key. To restore the entire hard disk, the command is

RESTORE A: C:\ /S /P

This command directs RESTORE to find the backup files on drive A and place these files on drive C, starting with the root directory (C:\) and traversing the entire directory system (*/S*). The program also queries whether any files on drive C that are marked read-only or archived should be replaced (*/P*).

If the hard disk is empty, the */P* switch is unneeded. However, the hard disk should never be empty. A freshly formatted hard disk should have the operating system placed on it first (through **FORMAT /S** or the combination of **SYS C:** and **COPY A:COMMAND.COM C:**). Because you do not want to replace the operating system files, you should give the */P* switch with the RESTORE command.

To restore only one or a few files, you can enter a file name that covers only the wanted files. The task is simpler when only one or two similarly named files are to be restored and no other file falls within the given name. For example, to restore FINPLN02.WK1 you issue the command

RESTORE A: C:\123\SALES\FINPLN02.WK1

Because the command line contains the specific path and file names, RESTORE searches the backup diskettes for the specific file. Only that file is restored.

To restore all worksheet files for the SALES directory, you use the command

RESTORE A: C:\123\SALES

Because a path but no file name is given, RESTORE assumes a file name of *.*. The command directs RESTORE to process all files backed up from the SALES subdirectory.

The problem arises when you attempt to RESTORE two or more files in the same directory or files in different subdirectories. For example, suppose that you want to restore FINPLN02.WK1 and FINPLN04.WK1 to your SALES subdirectory. The file name FINPLN*.WK1 restores these files but also overwrites any other files with the same basic name, such as FINPLN03.WK1.

If and *only if* you know that the most recent versions of all files covered by the given file name are on the backup set, you can give the RESTORE command without the */P* switch. For example, suppose that you know that the archive-bit incremental backup has all your updated WK1 files from the SALES directory. You use the command

RESTORE A: C:\123\SALES\FINPLN*.WK1

The needed worksheet files are restored. However, files that need not be restored also are restored. Time-consuming, unnecessary work is performed.

If you know that out-of-date versions of the file on the backup will replace recent versions, you should use the */P* switch. The command is

RESTORE A: C:\123\SALES\FINPLN*.WK1 /P

The */P* switch causes RESTORE to query whether to replace any file marked read-only or with the archive attribute turned on. You have the opportunity to answer **N** for any file that should not be replaced.

Suppose that you cannot give a restrictive file name to RESTORE because you have too many files to restore or the files are in too many directories. One trick is to use ATTRIB or a change-mode program like SDACHMD. You manually turn on the archive attribute for the files you do not wish to restore. You then use the */P* switch with RESTORE. As a result, RESTORE queries you on all the files you do not wish processed. Only the files with the archive attribute turned off are replaced.

The drawback to this trick is that your next use of BACKUP may back up too many files. You must manually clear the archive attribute on the files whose attribute was off before you ran ATTRIB or the change-mode program. You can simply reverse the commands you used to turn on the file attributes; that is, you can use *-A* or */A-* instead of *+A* or */A+*. You also may print a copy of the file names

and the file attributes by using an extended directory command (such as SDADIR) or file attribute program (DOS's ATTRIB or Norton's FileAttribute program). You can use this list to reset the attributes. If the task of restoring the attributes to their original settings is difficult, the trick is less useful.

Another point to remember about RESTORE and full file names is that some part of the full file name must be given. Unless you include at least a drive name or a file name, RESTORE does not function properly. For example, if you are attempting to restore files to the current disk drive and directory, neither of the following commands works:

RESTORE A: /P
RESTORE A: /S

RESTORE displays a warning message:

```
WARNING! No files were found to restore
```

The files are on the backup diskettes. RESTORE simply becomes confused by the lack of a destination name and the switches. To restore to the current directory when giving a switch, you use either of these two commands:

RESTORE A: C: /P

or

RESTORE A: . /S

Interestingly, if no switch is given, RESTORE works properly without a destination name, as in

RESTORE A:

To avoid confusion and wasted work, always give part of the full file name to RESTORE.

Using XCOPY: The Compromise Command

One of the new commands in DOS V3.2 is XCOPY, the copy command with a little extra. XCOPY is a compromise between COPY and BACKUP. Like COPY, XCOPY makes exact duplicates that do not need to be specially processed by RESTORE to be usable, and XCOPY does not erase the files already on the target disk. Like BACKUP, XCOPY copies only files that have changed since the last backup or since a certain date, and automatically duplicates the

source disk's subdirectory structure. (See Chapter 10 for more information.)

XCOPY goes above and beyond making backups, however. Unlike either COPY or BACKUP, XCOPY offers additional convenient options. XCOPY can

1. Copy only files that have changed since the last backup and *not* reset the archive bit
2. Prompt you before copying each file
3. Verify that the data is written correctly on the target disk
4. Pause for you to insert diskettes before the program begins its search for files to copy

XCOPY works faster than COPY or BACKUP because XCOPY uses your machine's RAM to read in as many files as possible and then write them out to the target disk.

The syntax for the XCOPY command is

dc:pathc **XCOPY** *ds:paths\filenames.exts dd:pathd\filenamed.extd /A /D /E /M /P /S /V /W*

The switches are listed and described fully in table 10.4.

XCOPY is a chameleon command, which can be made to look more like COPY one time and BACKUP the next. XCOPY can be used as a partial replacement for BACKUP or as an independent operation. For example, with the */A* option, XCOPY can be executed without interfering with your usual backup procedures. With the */M* option, XCOPY can substitute for some of the same backup procedures. More specifically, the */M* switch grabs files with the archive bit on and turns the bit off. Thus, this option copies files that have been created or modified since the last XCOPY /M or BACKUP /M was issued.

With the */S* and */E* switches, XCOPY can be ordered to re-create a subdirectory structure "on the fly" without disturbing files already stored on the target diskettes. The one characteristic of BACKUP that XCOPY does not mimic is the capability to copy a file that is larger than the total target disk size. Another lack is that XCOPY does not recognize hidden files.

The XCOPY command offers much additional flexibility in copying but also presents some possible stumbling blocks for your backup procedure. For instance, the DOS V3.2 manual states that XCOPY can be used to copy several subdirectories or an entire disk to a set of diskettes as long as no one file is larger than a diskette. The manual then suggests that you can easily make this a simple but

exhaustive procedure by executing the XCOPY command with the */S* and */M* options. When the target diskette fills up, you will see the "Disk full" message. If you issue exactly the same XCOPY command again, DOS ignores the files it has already copied (because the archive bit has now been turned off) and starts with the next file in the desired set.

This method is handy when you wish to transfer a large set of files (perhaps to another user). The main drawback is that you can poke large holes in your carefully planned backup strategy. If you back up on diskettes using the first routine outlined, your reliance on the */M* option of BACKUP for an incremental backup is shot to pieces. The files you copy with XCOPY /M will not be copied with BACKUP /M.

In this case, you probably should avoid the XCOPY /M option unless you use it as a direct substitute for one or more of your incremental backups. Just remember that the files backed up with XCOPY do not need to have RESTORE used on them; they are directly readable by DOS. If you routinely do your incremental backups with the */D* option, of course, the archive bit option of XCOPY has no effect.

Summary

This chapter explores many of the topics and techniques for backing up and restoring the hard disk, including general guidelines and several specific programs.

Remember these general tips. Perform master backups every month or couple of months to capture the entire hard disk. Perform incremental backups on a frequent basis to capture files you often change.

If you have only a handful of files to back up, DOS's COPY or BAC (from *PC Magazine)* works. For wholesale backup, DOS's BACKUP and RESTORE programs are required. BACKUP and RESTORE traverse the subdirectory system to capture files from anywhere on the hard disk. BACKUP offers the functionality to capture only changed files through the DOS archive attribute (*/M*) or through the file's date in the directory (*/D*). This capability makes incremental backups faster and easier.

XCOPY, which can operate like BACKUP, can be used for smaller backups or to transport sets of files between machines. The second use can keep office and home computers or office and laptop computers in sync.

Chapter 16

Backup Alternatives

Backup is a tiring chore. Many people skip the process because it takes too long, is inflexible, or requires too much work. But the backup process must be executed on a faithful timetable because skipping a backup jeopardizes your files.

The BACKUP and RESTORE combination makes a powerful team but offers few savings in time or flexibility. Simply stating that backing up a 20M disk drive takes forty-five minutes to an hour and a half lacks impact; the words don't adequately describe the tedium. You must experience the burden yourself.

Fortunately, you have alternatives—both hardware and hardware-software combinations—that greatly speed the backup process. These alternatives give better options for file selection or exclusion and make the task easier and more convenient.

Software Backup Programs

Programs are available to make backup more than manageable. Watching the names of your directories or files flicker across your screen as one-by-one they are backed up in a matter of seconds is truly impressive.

This section describes three of the many software-only contenders: FastBack™, BackTrack™, and DSBackup™.

FastBack

The real backup speed demon is FastBack by Fifth Generation Systems. Speed, speed, speed (and some improved flexibility) are FastBack's advantages. FastBack performs its magic by using the direct memory addressing (DMA) circuitry of the computer and by dividing the backup job into several processes, each accomplishing one specific task.

FastBack consists of a set of programs: FASTBACK.EXE, FRESTORE.EXE, and a test-and-install-program. The testing phase of the installation program ensures that your computer's circuitry can be used successfully. The program also "installs" the disk drives to be used for the backup diskettes. If you have two floppy disk drives, FastBack uses both. However, if you have two disk drives of different capacities, you should use only the larger drive.

FastBack accepts different kinds of information about the files to be backed up. You supply the disk drive, starting directory, and file names (as three separate items). You choose whether the program should traverse directories and whether it should back up only files that have been changed (based on the archive attribute). You can enter all this information on the command line. If you supply insufficient information or none on the command line, the program prompts you for each item.

An added feature of FastBack is the use of a command file similar to a batch file. Through the command file, you can specify more than one full file name so that you can issue a single command to back up several files or directories with dissimilar names. This feature surpasses DOS's BACKUP, which allows only one full file name. This capability is also convenient for system administrators, who can establish batch and command files for infrequent or novice users. One command (the batch file) runs all (FastBack with the command file).

FastBack has some compromises. The program backs up only hard disks to floppy diskettes. FastBack files and diskettes are not readable by DOS, but unreadable files are not really a problem. Just as BACKUP's files must be restored with RESTORE before they can be used, FastBack's files must be restored with FRESTORE.

The inability of DOS to read the diskette might cause a problem if you mistake a FastBack diskette for an unformatted diskette.

Reformatting that diskette could court tragedy. This misfortune is possible if you violate the practice of storing backup diskettes in a group and in a safe place.

FastBack cannot back up by date. The program has no equivalent to the */D:date* option of BACKUP. Considering the speed of FastBack, this disadvantage is very minor.

The first versions of FastBack were copy-protected by the key disk scheme (versions after V5.03 are not), but the restoration program, FRESTORE.EXE, was never copy-protected. If you lost your FastBack diskette, you could restore previously created backups.

FastBack offers a number of advantages. The software requires no additional hardware. Formatted diskettes are *not* needed because FastBack formats diskettes as it runs. In fact, FastBack reformats formatted diskettes according to its own formatting scheme. You don't need to worry about running out of formatted diskettes in the middle of a backup.

But the principal advantage of FastBack is speed. You can back up an unmodified 6 MHz Personal Computer AT's full 20M disk in 19 minutes with unformatted diskettes, less than 12 minutes with previously FastBack-formatted diskettes. Including formatting the diskette under DOS, the same operation with BACKUP under ideal conditions takes about one and a half hours. FastBack is seven times faster.

The first time you run FastBack, the speed is almost disconcerting. File names fly across the screen. You also must become accustomed to removing and inserting floppy diskettes *while* the red light is on (the disk drive is spinning), something you have been trained to avoid under pain of death.

Actually, with FastBack, changing diskettes with the disk drive spinning is not only acceptable but necessary. FastBack gains some of its speed advantage by keeping the floppy disk drive motor running instead of shutting down and then spinning back to the correct speed. On some computers unlike the AT, you press a key after changing diskettes. On an AT or similar computer, you just close the door, and FastBack continues (see fig. 16.1).

If you have two floppy disk drives available, FastBack uses both, thus decreasing the delay for changing diskettes. You change the diskette in the inactive disk drive while FastBack continues with the other. You actually play "beat the clock" as FastBack displays the amount of time the program waits for diskettes to be changed—a figure your conscience does not allow to go very high. When the backup is

```
Fifth Generation Systems, Inc                                  07B06752 v5.13
7942 Picardy Ave., Suite B350      F A S T B A C K (tm)        (c) 1984, 1985
Baton Rouge, LA  504-767-0075                                  all rights reserved

Elapsed time 00:21 | 0 secs operator | ESC to exit | DOS performance: 25

CHANGE DISKS Drive A:
  Track 12   Drive B: Disk # 1   3-Aug-86 16:55

C:\BACKUP.M_U
```

Fig. 16.1. *The FastBack prompt to change disks.*

finished, FastBack displays complete statistics about the backup job, such as total time, time per K of files, and other figures (see fig. 16.2).

```
Fifth Generation Systems, Inc                                  version 5.03
7942 Picardy Ave., Suite B350      F A S T B A C K (tm)        (c) 1984, 1985
Baton Rouge, LA  504-767-0075                                  all rights reserved

Backup is contained on 1 disks (#1 - #1)

1 minutes, 4 seconds to copy 67584 bytes in 6 files
   1.063 kbytes/second (avg)
   0.064 Mbytes/min (avg)
  14.000 Seconds to change disks (avg)
   6.000 files/diskette (avg)
   67584 bytes/diskette (avg)

Press (return) \
               \
```

Fig. 16.2. *FastBack summary statistics.*

The FRESTORE program handles file restoration and verification. You select to verify or restore one file or a group of files as the file names are shown on the screen. Unfortunately, you cannot easily

select a set of unrelated files (as you can with the FASTBACK.EXE command file). Selection, however, is very fast.

To obtain this high selection speed, FastBack creates a catalog of the backed up files if the subdirectory FASTBACK exists. Using the catalog to choose selected files is fast. The process does have one disadvantage when you make subsequent backups. If FastBack finds the catalog (FASTBACK.CAT), the program assumes that you wish to append files to your current backup diskettes. To make a new backup set, you must either delete or, preferably, rename the FASTBACK.CAT file (using the month, day, and year as part of the file name). When you again wish to use FRESTORE on a backup set, you change the name back to FASTBACK.CAT.

For files previously existing on the hard disk, FRESTORE can be instructed to query each replacement for verification, to replace the file with the backup version without query, or to skip the file. A lack in FRESTORE is that the program does not query you whether to replace a read-only file (similar to RESTORE /P). The program does ask about replacing a more recent file with an out-of-date backup, but you must remember which files are read-only and should not be replaced.

The brief but thorough FastBack documentation points out a factor that greatly affects the speed of operation. Large subdirectories (with several hundred files) slow FastBack considerably. Limiting directories to fewer than 100 to 150 files should not be considered a limitation; this restriction is good operating practice.

FastBack operates so quickly that full hard disk backups are not a chore. In fact, the process is so quick that you can omit incremental backups, which are made because master backups take so long, and always make master backups. With a total backup time of less than 20 minutes for a 20M hard disk, the backup chore becomes convenient with FastBack.

For backup integrity, FastBack uses a proprietary error-checking and correcting scheme to ensure that the files can be recovered intact. Fifth Generation boasts that a diskette can be stapled and no data will be lost. One computer columnist tried this. His only complaint was that the program could not remove the staple. All files were recovered intact.

When the program is installed, FASTBACK.EXE and FRESTORE.EXE are created in the FASTBACK directory. You can run the programs from this directory, or you can move the two files to the BIN or UTILS directory. If you create command or batch files for

the program, you should keep the command files in the FASTBACK directory and place the batch files in the BIN or UTILS directory.

BackTrack

Backing up is a deliberate process. You stop your work, attend to the backup, and then may return to your work. This statement is not necessarily true if you use BackTrack, a program from Tallgrass Technologies.

BackTrack is a resident program, similar in concept to SideKick. BackTrack does not make appointments for you, but it makes appointments for your files. At a set time or when awakened by the computer's inactivity, BackTrack automatically backs up any changed files on your hard disk. BackTrack can use floppy diskettes, another fixed disk, a removable disk cartridge system, or—for BackTrack V2 or later—a Tallgrass PC/T tape drive. Once loaded, BackTrack requires no operator attention except when removable media must be changed. The backup process is actually faster than the BACKUP command. The operator does not select files. BackTrack automatically selects files based on the archive attribute. The entire process is automatic, and backup is painlessly performed.

BackTrack does its job when the computer has been unused for a user-determined length of time, called a *lull*. If even the planned interruption of your train of thought may be slightly unsettling, you can cause BackTrack to wait for another lull simply by pressing a key. If you find automatic anything during your worktime upsetting, you can set BackTrack to start at specific times of the day. This feature is helpful if you use the system in spurts and do not wish to be disturbed. This feature is also helpful when you know the times your machine will be unattended, for example, during a coffee or lunch break. If you finish your day's work early, you can override the time settings and direct BackTrack to finish all pending backups immediately.

The program gives you the option to devote a drive exclusively to the backup operation or to share the disk drive with other programs. If you choose the latter, BackTrack double-checks that the diskette in the drive is the right one.

Another option available is *simultaneous* operation, in which BackTrack steals idle time to perform the backup operation. This mode can be used when the computer is subject to concentrated use without rest. This mode could help those in a computer-hostile environment (frequent power outages) or when even a few hours'

work cannot be lost. The simultaneous mode is less satisfying than the other two modes because you and BackTrack contend for the computer's resources, and therefore the operations are slower.

In addition to incremental backup routines, BackTrack also can make a master backup, which is recommended monthly. Additional functions provide estimates of the number of diskettes and time required to finish the backup, list files waiting for backup, reuse old backup diskettes, and format new diskettes.

BackTrack can work with floppy diskettes, but using diskettes can be awkward. You must manually change the diskette when it is filled. If the disk drive is unattended, the backup waits. A floppy disk drive or hard disk drive used as the backup device should be dedicated solely to BackTrack's use. When you use a tape cartridge backup, which typically holds 10M to 60M of information, BackTrack achieves its full potential for automatic backup convenience. Regardless of whether you purchase a tape drive or reserve a floppy disk drive or hard disk drive for BackTrack, the price may be small compared to the convenience of truly automatic backups.

DSBackup

DSBackup from Design Software is an excellent utility for quickly and painlessly backing up and restoring files. The program offers excellent on-line help, and the operations are so clear and simple that reading the manual is almost unnecessary. The only pointers needed from the documentation are the default options for the restore program and the procedure to *exclude* files or groups of files from processing.

An unorthodox convention that must be remembered is that DSBackup reverses terminology for its restoration program. For backup, the source is the disk and directory of the files to be backed up. For restoration, the source is the disk and directory where the backup files should be placed. Design Software's restore syntax is backward compared to all other backup utilities. For example, when you back up your entire hard disk, drive C, to drive A, C:\ is the source and A is the destination. When you are restoring files, C:\ is still the source, although all other programs would consider C:\ to be the destination. To the company's credit, Design Software tries to make this convention very clear.

When you first boot the program, you see a simple four-window display (see fig. 16.3). At the top is the designation of the source

and target directories. On the right is the basic menu, and on the left are some statistics and the options' status. In the bottom window are displayed names of the files being considered for backup.

```
DSBackup Version 2.4e, (c)Copyright 1985, 1986 by Design Software

    Source Directory : C:\
Destination Directory : A:\

          Time : 05 01pm                    Toggle Menu        Filespec(s)
          Date : 08/03/1986                 Source Dir         Destination Dir
   Floppy Disks : 37                        Backup             Restore
     Total Size : 5740K                     Execute FORMAT     Format Location
         Verify : Off                       SAVE setup         LOAD setup
Subdirectories : ON                         HELP               Exit
Prompting dirs : NO prompt
   Backup type : NORMAL

   BACKUP   M_U   CONFIG   TOP   CONFIG61 DRI   CONFIG   PM3   NOTES    ARC
   AUTOEXEC OLD   ARC2     99    CONFIG   SYS   REPORT   DBU   LIGHTEST BAK
   CONFIG   BAK   LIGHTEST BAT   AUTOEXEC BAK   NOTES    USQ   NOTES    LU
   PCTOOLS  OVL   AUTOEXEC BAT   WIN100   OVL   WIN      INI   WINOLDAP MOD
   WINOLDAP GRB   WIN100   BIN   COURA    FON   HELVA    FON   TMSRA    FON
   ROMAN    FON   SCRIPT   FON   MODERN   FON   EPSON    DRV   COURD    FON
   HELVD    FON   TMSRD    FON   COURC    FON   HELVC    FON   TMSRC    FON
   ABC      TXT   DOTHIS   TXT   README   DOC   PRACTICE DOC   WSTEST   DOC

Cursor to Choose Item, Space Bar to Execute, ? for HELP
```

Fig. 16.3. *Initial display of DSBackup.*

The selection criteria is broad. You can include files for backup by selecting

- Specific subdirectories
- All subdirectories of a specific directory
- Any DOS file attribute (archive, hidden, system, and read-only)
- Any combination of the preceding

Although you can give up to 16 full file names, 16 file names may not always be sufficient for selective backup. DSBackup extends the selection process by allowing you to include or exclude files that match any of the 16 given file names. When you choose the "Filespec(s)" option, the program opens another window in which you enter the full file names, which can include wild-card characters. If you place a minus sign (-) in front of a name, DSBackup will not process files that match this file name (see fig. 16.4). For example, you can exclude all executable files by specifying

-*.COM
-*.EXE

```
DSBackup Version 2.4e, (c)Copyright 1985, 1986 by Design Software

    Source Directory : C:\
Destination Directory : A:\

        Time : 05:00pm                  1 *.*          9
        Date : 08/03/1986               2 -*.COM      10
 Floppy Disks : 2                       3 -*.EXE      11
  Total Size : 160K                     4             12
      Verify : Off                      5             13
Subdirectories : Off                    6             14
Prompting dirs : NO prompt              7             15
  Backup type : NORMAL                  8             16

  BACKUP   M_U   CONFIG   TOP   CONFIG61 DRI   CONFIG   PM3   NOTES    ARC
  AUTOEXEC OLD   ARC2     99    CONFIG   SYS   REPORT   DBU   LIGHTEST BAK
  CONFIG   BAK   LIGHTEST BAT   AUTOEXEC BAK   NOTES    USQ   NOTES    LU
  PCTOOLS  OUL   AUTOEXEC BAT

Use cursor keys to move, SPACE to change/edit, ESC to start search
```

Fig. 16.4. *DSBackup "Filespec" window.*

Unfortunately, the documentation does not state clearly that when you exclude files, you must give at least one *inclusion*. I made several attempts before discovering that something must be included before different files can be excluded. When all else failed, I entered the *.* file name to include all files; then I could exclude a group of files.

When backing up the files based on the archive attribute, you can direct DSBackup to clear the bit or leave the bit set. This option helps when you are performing two different backups: for example, a smaller backup to capture certain files and then a major backup to capture all created or changed files.

During the restoration process, DSBackup does query you on the subdirectories to restore, the only prompting the restoration process offers. Fine-tuning can be achieved by using the 16-file-specification feature. Accepting 16 full file names for inclusion and exclusion covers all common and most possible situations.

As with any backup program, good operations practice dictates that you number the backup diskettes. With DSBackup, noting the subdirectories backed up on each diskette is also helpful. Unlike BACKUP and FastBack, DSBackup can restore files to a subdirectory different from their original subdirectory. This feature helps when you want to relocate files, but the feature also can confuse unwary users, and files can end up in the wrong directories. The program

provides some aid by documenting, to a disk file or the printer, the contents of the entire backup set. When you are restoring files, careful operation is still required.

DSBackup gracefully handles the problem of running out of or forgetting to format diskettes. Before starting the actual backup processing, the program displays the number of floppy diskettes needed. When you change the selection criteria, the diskette calculation is updated. When another diskette in a set is required, DSBackup prompts

```
Continue backup
Format Destination Drive
Cancel Backup
```

If the DOS FORMAT program is on the current source disk drive or DOS PATH, you can format a diskette and resume DSBackup without restarting. DSBackup's handling of unformatted diskettes is a welcome improvement over DOS BACKUP's all-or-nothing attitude.

The information for DSBackup can be entered on the command line. The program prompts for any missing information. The following command line directs DSBackup to perform a complete backup of drive C (the source—S) to drive B (the destination—D), to include all subdirectories, and to display prompts:

DSBACKUP S=C:\ D=B:\ SUBS PROMPT BACKUP

As with FastBack, you can create one or more "load files" (command files) that tell DSBackup the files and options to use. You can direct DSBackup to the load file from the command line or after the program has been invoked.

DSBackup is fast; I backed up about 8.5 megabytes in 6 1/2 minutes on a plain PC with the file-verification option on. The visual display is impressive as each file being backed up is highlighted in the bottom window and then "flies" off the screen.

One reason for DSBackup's speed is its file map. Before beginning the actual backup, DSBackup scans all directories for the files to back up and builds a map (list) of the eligible files. This preliminary scan is faster than an "on the fly" examination of files to process. After completing the backup, the program writes this list on the last backup diskette. The list guides the restoration process so that files are located quickly and restored. DSBackup also places all files in one large "file," which can span several diskettes.

The DSBackup manual is easy to read and well developed. The manual begins with a "quick start" introduction, followed by a

demonstration of typical operations (a complete hard disk backup, the complete restore, a selective backup, and a selective restore), and by a complete reference section.

You can locate the program any place on your hard disk. The installation program automatically creates its own directory.

Tape Drives for Backup

If playing the floppy shuffle—even with 1.2M diskettes—seems tedious, if time is at a premium, or if you have the mammoth variety of hard disk (30M or more), software-based backups may not be entirely appealing.

Hardware alternatives are indicated, especially with the increasing number of choices and lower prices. The idea of safely storing 20 megabytes on a 2 1/2-by-3 1/2-inch tape cartridge can be attractive.

Although the common temptation is to choose the fastest unit, speed is only one of many major considerations. Other important factors are compatibility, reliability, and ease of use.

As with any other equipment, you should question whether the prospective units are compatible with your computer, even with IBM or compatible machines. Some units that rely on specifics within IBM's version of MS-DOS (PC DOS) may not function correctly with a compatible. Most will, but a few won't. The best test is personally witnessing the particular tape drive working with your specific configuration.

Because backups cannot be compromised, the reliability of the tape backup unit cannot be compromised. The discovery that your unit does not work will normally occur after you need to restore the files. Then the discovery is simply too late.

Carefully evaluate the reputation of the tape drive unit and its manufacturer by consulting friends, business associates, and computer periodicals. This selection is as important as your initial hard disk selection—maybe more so. Getting the best deal (price) from a hole-in-the-wall dealer with brand X equipment may be the worst deal in terms of reliability and service. If you must compromise on price, do so by conceding speed. Don't sacrifice reliability.

The Software

Another important factor concerns the software that comes with the unit. Evaluate the ease of operation and ease of use. How

thorough are the options offered? Can you understand the program without extensive help? How convenient is the program to run?

In the old days of computer history, tape backup was a simple and fast process. The operators performed an *image* backup, a sequential byte-for-byte snapshot of everything on the hard disk. Image backups have two advantages. The image backup is quicker than a file-by-file backup, and the software for the backup is less complex and easier to write.

Image backup has two major disadvantages. Selective file backup is almost impossible. Restoring less than the entire hard disk ranges from difficult to exceptionally time-consuming. Image backup processes contiguous sectors of the hard disk. As you recall from previous discussions of file fragmentation, your files become fragmented as you use your disk. Fewer files are stored on consecutive sectors. Attempting to back up selected files by backing up consecutive sectors becomes increasingly difficult.

Restoring selected files is an even more difficult task. You must know "where" the file was stored to restore the specific sectors that held the file. Even if you can identify the correct sectors, another factor stands in your way. If files that have changed since the last image backup sit in the sector that must be restored, you must first copy these files to some media, then restore the sectors, then recopy the files back to the hard disk.

In the worst case, you must make an image backup of the entire disk, restore a previous image that holds the files to be restored, copy these files to a removable media, again restore the entire disk using the image backup you just made, then restore the files you temporarily copied to the removable media. Understating the issue, this process lacks convenience.

Times have changed. The ever-increasing population of hard disks and growing sophistication of hard disk owners have created a need for more flexible and powerful tape backup and restoration capabilities.

The software with the unit should provide three types of backup modes: total image backup, total file-by-file backup, and selective file-by-file backup. Although total image backup is the worst to use when you need to restore less than the entire hard disk, the process is faster than total file-by-file backup. Total file-by-file backup is still a full backup but gives you the option of restoring only selected files. Selective file-by-file backup is speediest when backing up and also offers the greatest speed and flexibility when restoring files. Generally, any program handling a file-by-file backup performs both

total and selective backup and is sufficient for most backup needs. Having all three modes available might be beneficial, however, depending on your disk capacity and backup inclinations.

If possible, therefore, examine the tape drive's software. Depending on the user of the software (neophyte or experienced), a menu-driven (select by number), a command-line (enter all options on the command line), or command-file (all responses in a disk-based file) program may be best. The menu-driven and command-file programs are best for newcomers. (An experienced user can establish the command files for the neophyte.) The command-line and command-file programs are better for more proficient users. Having all three modes is best, but having the first and second is sufficient because a batch file can do the work of a command file.

The Tape Units

Most tape units today use 1/4-inch magnetic tape in cartridges or cassettes. The standards for recording information, however, are developing slowly. The three dominant standards (obviously having only one standard would be best) are the QIC standard interface developed by the Quarter-Inch Cartridge committee, a group of manufacturers from the tape industry; the PC/T standard (Personal Computer/Tape), introduced by Tallgrass Technologies; and the quasi-"floppy tape" standard, so named because these units are connected to your floppy disk controller card. Not surprisingly, IBM uses a fourth standard, a modified form of the QIC standard.

Although standards are helpful when you are interchanging cartridges between computers, the important use of the cartridge is for backup. In this case, the cartridge is not used on other computers. Standards have almost no meaning if the sole use of the tape unit is for backup. The important issue relates to compatibility. Will the tape drive work with my computer? The "standard" used by the tape drive is almost a moot point.

The most common tape cartridges available today are DC300, DC600, DC1000, and DC2000. These cartridges use 1/4-inch or 1/8-inch (technically .15 inch) tape. The information given here is correct at this writing, but tape and tape drive manufacturers continue their work to pack information more densely and so increase the tape's capacity.

DC300, DC600:	Quarter-inch tape cartridges about the size of a videocassette (4 by 6 by 5/8 inches). The DC300 is the older of the two and holds less tape than the DC600. The storage capacities are 30 megabytes for the DC300, 60 megabytes for the DC600.
DC1000:	One-eighth-inch minicartridge tapes using the same general construction as DC600. The size of the DC600 approximates that of an audiocassette (2 by 3 by 1/2 inches). The capacity is 10 megabytes.
DC2000:	Same cartridge size as DC1000 but using quarter-inch tape. The capacity is 20 megabytes.

Digital cassettes, used in some tape systems, follow almost no standards for recording information. Most digital cassettes look generally like an audiocassette but vary in specific form (placement of write-protect tabs, capstans, and so on). The cassettes usually hold less information than cartridges. Cassettes are less sturdy than cartridges, which have a more rigid plastic casing and a metal base plate. For these reasons, tape cassettes are less popular than tape cartridges.

More Tape Drive Considerations

When you are selecting a tape drive, most of the issues you examined when selecting your hard disk arise again: external unit versus internal, available expansion slots for the controller, sufficient power supply, price versus performance, reliability, and repair.

One practical consideration is whether to purchase an external or internal tape unit. Internal tape drive units conserve desktop space, sometimes a commodity more precious than gold. However, does the computer have the room to house an internal unit? Some half-height tape drives, because of the controller card mounted on the disk drive, may be slightly larger than half-height disk drives.

Expansion slot space may be an issue. Some units, internal and external, require an expansion slot-and-a-half or two full slots for the adapter or controller card. Some units use only one expansion slot. Some units do not use a slot at all. Instead, these units use the spare connector on the back of PC's and PC XT's floppy diskette adapter. However, these units are considerably slower. If you have enough

slots, the size of the adapter/controller card is a lesser issue. Having limited slots or no slots narrows the potential range of units.

Most tape drives use little power. Unless you have power-hungry expansion boards and disk drives, the power supplies of the PC XT and AT are sufficient. A hard disk drive and a tape drive exceed the capacity of the PC power supply, however. If you have upgraded the power supply of the PC to 115 watts or more, the tape drive and hard disk drive should work.

As with hard disk drives, tape drives undergo many of the same prepurchase questions. If internal room, power, and expansion slots are sufficient, an internal tape drive can be used. If any of the preceding are insufficient, an external tape drive should be used. If available expansion slots are limited or nonexistent, you must give careful consideration to the single card or an external tape drive.

A Tape Drive Example: The Irwin 310

Irwin Magnetics, like most manufacturers, produces a spectrum of both internal and external tape drives in a variety of sizes and capacities. I use an Irwin 310 BACKUP™ Tape Subsystem on my PC.

The 310 is an external unit that has its own power supply, so the tape drive does not burden the power supply of the PC. The 310 is a "no-boarder." It does not require a separate adapter card because the unit plugs into the back connector of the PC's floppy disk controller. The tape drive box itself is about 7 inches wide, 12 inches deep, and a little more than 4 inches high. The unit has only one "control," a little push bar, which is pressed to eject the cartridge (see fig. 16.5).

The unit uses DC1000 10-megabyte cartridges. As with all tape cartridges, a write-protect switch is built into the media. Loading a cartridge is a simple matter. You push the cartridge in the drive's front door until you hear the reassuring click, which indicates that the cartridge is seated properly.

These units take approximately 18 minutes for a complete image backup with verification. The file-by-file backup of the same 8.5 megabytes takes 19 minutes. These backup times are not the speediest; they place the unit in the low-to-middle range for speed.

My worst disaster with the tape drive is humorous in hindsight. Several weeks after I had first used the drive, I made a silly mistake that taught me an important lesson. When taking a cartridge from its container, I also unknowingly grabbed the loose piece of paper containing peel-off labels, which are provided with the tape. The paper is the same size as the cartridge and fits perfectly under the

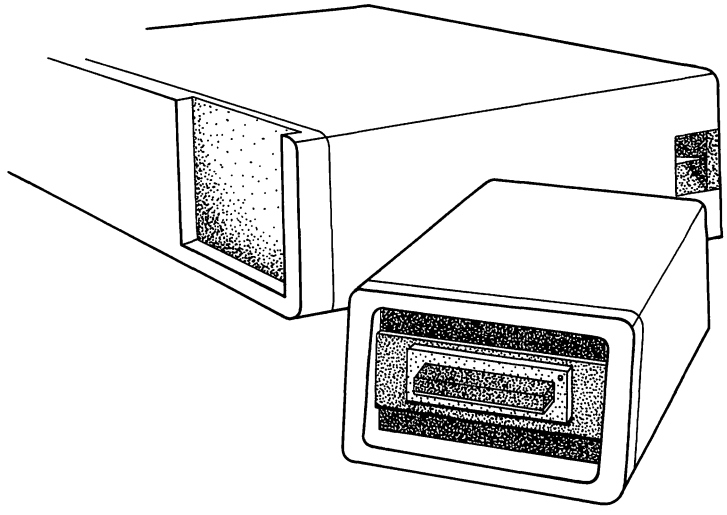


Fig. 16.5. *The Irwin 310 tape drive box.*

metal plate of the cartridge. I stuck the cartridge-paper combination into the tape unit and the unit started. It would still be running today if I hadn't pushed the tape-eject bar.

I noticed the piece of paper but at first didn't realize what had happened. On the bottom of the tape cartridge is a small window. Through this clear window, a light shines on a sensor in the drive. The sensor uses the light to detect when the end of tape is reached. With the paper covering the window, the little Irwin kept looking for the end of the tape, which had spun off the end of the reel long ago.

Apparently, this problem is a common occurrence. Irwin takes pains to warn about that pesky paper in the documentation. The warning is all Irwin can do because it does not manufacture tapes. The solution is to use the labels immediately and pitch that paper the first time you open the tape container.

Installation

Physical installation is simple. The Irwin 310 runs off the PC's floppy disk controller card. The first step is plugging the single cable from the tape drive into the "external disk" connector on the back of the PC or PC XT. (ATs require a special board for the connection because the disk adapter does not have this external connector.) The last step is to plug the power cord into a wall socket.

The software installation consists of copying three files to your hard disk and modifying your CONFIG.SYS file. The FILES and BUFFERS lines must be included with the following minimum parameters:

FILES = 10
BUFFERS = 20

Most users already have these lines in their CONFIG.SYS files. If the lines have higher values, the speed of the backup-restore operation is boosted. To use the unit, you reboot the system and switch on the tape drive.

The Software

The tape software consists of three programs: TFORMAT, IMAGE, and FIP. The TFORMAT program prepares a tape to receive data. Ordinarily, TFORMAT is run only once on the tape, just as FORMAT is run once on a diskette. IMAGE and FIP perform the actual backup and restore procedures.

Tape formatting involves two operations. *Servo-writing* places a pattern of signals on the tape to guide the read/write head precisely. *Formatting* lays out the magnetic pattern of sectors, blocks, and tracks and identifies any defective areas. Together, these two steps take about 35 minutes. The TFORMAT program can also examine the cartridge for proper format and for data so that a cartridge with useful information is not reformatted.

To make a complete image backup of the hard disk, you use the IMAGE program. The menu is simple; you can back up everything, restore everything, or quit (see fig. 16.6). An image backup overwrites any data already on the tape. To prevent accidental erasures, IMAGE checks the parameters of the backup tape; if the cartridge has been previously used, IMAGE asks whether you wish to continue or switch cartridges. If your stored files exceed the capacity of a cartridge, IMAGE splits the backup across multiple cartridges.

The restoration process is also straightforward. Remember that restoring from tape destroys any data currently on the target—the hard disk. Image restorations are most useful when the entire disk must be restored and are almost useless when you need to restore only one or two files. One advantage of an IMAGE backup and restore is that files are unfragmented. The program restores the files in contiguous sectors regardless of where they were stored previously.

```
Image Backup Program     Release 4.2   (C)1984,85,86 Irwin Magnetic Systems
Pre-Release Beta Version, Not Intended For Production    Ser. No. 140342017

                          IMAGE Menu

                    1. Image backup to tape
                    2. Image restore from tape
                    3. End image

Enter menu selection [3]:
```

Fig. 16.6. *The Image Backup menu.*

To back up selected files, you need the File Interchange Program (FIP). When you run FIP, the files are grouped into *save sets*; one set is created each time you back up. The DC1000 tape can hold up to 93 save sets, so tape cartridges may be used for more than one backup operation. Additional save sets—but not image backups—can be added to the tape.

You do not need to reformat the tape to discard old save sets. Setting the tape status to "unused" destroys the tape's directory, eliminating old save sets and allowing the tape to be reused.

FIP offers a variety of ways of selecting files for backup (see fig. 16.7). You can select certain directories, certain files, files created or changed on a given date, files created or changed on or after a given date, or files modified since the last backup. However, to execute more than one of these choices in sequence, you must run each menu option separately.

```
File Interchange Program   Release 4.2 (C)1984,85,86 Irwin Magnetic Systems
Pre-Release Beta Version, Not Intended For Production    Ser. No. 140342017

                          FIP Menu

  1. Backup all files by date and time        6. Restore an entire saveset
  2. Backup all files with a specific date    7. Restore directories
  3. Backup all "modified" files              8. Restore selected files
  4. Backup selected files
  5. Backup directories

                    9. List savesets on current cartridge
                   10. Set cartridge status to UNUSED
                   11. Enter FIP command structure
                   12. Exit program

Enter menu selection [12]:
```

Fig. 16.7. *The FIP menu.*

On the other hand, suppose that you need to specify a large and disparate group of files (a group not represented by the choices on the FIP menu). FIP also offers a command mode, in which you can specify a combination of file selection criteria. For example, you can specify that FIP should process all files in one subdirectory, only the DOC files in a second subdirectory, and the WR1 files in a third. The various selections can be placed in the same save set. You can bounce around from directory to directory, selecting only the files you want. You can display a list of the file names already selected, modify the list again, and continue the selection process until you have the set you wish to back up. Typing **BACKUP** then activates the tape backup.

The only option the FIP command language does not offer is the capability to save the backup selection criteria in a file that can be recalled later. The missing option is not a real hindrance.

As mentioned earlier, restoring a backup with IMAGE is straightforward but inflexible. The entire hard disk is restored, destroying all previous information. FIP's restoration options are more flexible. FIP can restore individual save sets, subdirectories processed using the "Backup Directories" option, or individual files within a save set. When you run FIP from the menu mode, files are restored to their original subdirectories. Like DOS's BACKUP, FIP creates missing subdirectories as needed. From the command mode, you have to change the default directory before you issue the RESTORE command.

Increasing the number of disk buffers speeds the FIP restore operation. If you have had some other reason to limit (or not specify at all) the number of disk buffers in your CONFIG.SYS file, a good technique is to modify temporarily the CONFIG.SYS file just before you restore. For the best speed during a restoration, the Irwin documentation recommends adding or modifying the disk buffers line to

BUFFERS = n

where

$$n = \frac{(\text{number of files} \times 32) + \text{average file size in bytes}}{512} + 5$$

Of course, you need to reboot your machine to put the new buffer setting into effect.

The Bernoulli Box: A Hard Disk/Tape Drive Alternative

Although not literally a hard disk drive, the Bernoulli Box, manufactured by Iomega, is for many users an appropriate substitute or supplement for a hard disk drive.

The Bernoulli Box is named after the eighteenth-century Swiss mathematician Daniel Bernoulli. He was the first to note and describe the aerodynamic principle that when the air above a surface moves faster than the air below that surface, the surface rises. Airplane wings and the Bernoulli media both use this principle for lift.

The Bernoulli Box uses a flexible material similar to a floppy diskette mounted on a shaft. When stationary, the media droops like a floppy diskette. When the media spins, the Bernoulli plate mounted above the media guides the air flow and increases the air flow speed, thereby lifting the disk toward the plate. The Bernoulli plate carefully controls the air flow so that the diskette flies microscopically close to the drive's recording head. Because the gap is kept very stable and very small, information is recorded densely on the diskette surface. Because of speed, storage capacity, and diskette rigidity while spinning (the media is called by some a *frigid* disk), the Bernoulli Box resembles a hard disk.

Iomega touts the Bernoulli Box as more expandable, more convenient, and safer than a hard disk. Each advantage is based on the Box's removable media or its unique construction.

In a sense, the Box has unlimited expansion for storage capacity. Like floppy diskettes, the cartridges can be changed. When a cartridge is filled, you simply pop in another cartridge. A Bernoulli cartridge holds 10 or 20 megabytes. Obviously, only so much storage can be on-line at one time. The maximum drive capacity is 20M, but dual drive units are available so that the on-line limitation is raised to 40M. Iomega also markets the Bernoulli Box Plus, a unit with an 80-megabyte fixed disk and two 20-megabyte Bernoulli drives.

The convenience of the Bernoulli Box is also due to the removable media. The cartridge is compact, just slightly larger than a typical diskette and weighs about one and one-half pounds. Several cartridges fit easily into a briefcase, allowing convenient transportation of hundreds of megabytes of information. For security, the cartridges can be removed at night and stored in a locked desk

drawer or safe. Even small fixed disk drives cannot easily be removed and so lack this convenience.

As for safety, the Bernoulli Box cannot suffer the feared head crash of conventional hard disks. With a hard disk system, any significant contamination or shock causes the recording heads to nose-dive into the platter. As mentioned many times before, the crash can damage your drive, your data, or both. The dynamics of the Box are radically different from those of typical hard disks. On the hard disk, the heads fly and dive into the platter. On the Bernoulli Box, the *medium* "flies." When a problem occurs, the medium flops and makes a landing *away* from the recording heads. Because the heads and media do not make contact, ruining data is impossible.

The major disadvantages of the Bernoulli Box are its price and the uniqueness of the technology. The Box is an expensive unit—a 20-megabyte dual drive retails at about $3,000. Dollar for dollar, the Box is almost two to three times more expensive than a fixed disk. However, a direct price-to-storage comparison is difficult. Hard disks have a fixed storage capacity. Like the floppy disk drive, the Bernoulli Box can store more than 40 megabytes if you change cartridges, which cost $50 to $60 retail. The Box is also its own backup device. You pay more for initial storage but get the added conveniences of expandable storage and a self-backup device.

Only Iomega uses the Bernoulli technology for a storage device. When a device is dependent on a single manufacturer, the prospective buyers rightly ask whether they might become orphans (because the company goes out of business or abandons the Box) or be subject to pricing whims of the manufacturer. The answer to both questions is "Yes, but highly unlikely." Although no one can predict the future, Iomega is a mature company dedicated to serving its customers.

The Bernoulli Box has spread in popularity enough to reach the "critical mass" of acceptance. Enough users have bought and use the device that Iomega has a stable customer base, which the company must service. Having been in business for several years, Iomega appears to be stable. Disk competition, which some would fear, actually helps the Bernoulli Box owner. Price increases from Iomega are kept in check by growing competition of other removable disk drive manufacturers and the shrinking prices of the fixed disk.

The Bernoulli Box is perfectly suitable for a PC, XT, or AT environment. Because the Box has its own built-in power supply, the unit does not tax the computer's power output, an important feature

for low-powered PCs and XTs. The unit can be a direct replacement for a hard disk or an add-on to a hard disk system.

Many AT owners, citing the reported problems of the original AT hard drives, have opted to replace their (original or never-purchased-in-the-first-place) hard disk with a Bernoulli Box. On the other hand, users who want to keep their hard disks buy a Bernoulli Box for the expandability and the backup potential. If you are considering the Bernoulli Box as a replacement for a hard disk, the dual drive unit makes the most sense. Like having two identical floppy disk drives, having the two-cartridge system allows you to back up cartridges onto other cartridges. The unit is a convenient backup device as well as the computer's primary storage.

Physical Installation

When you are installing the Bernoulli Box, possibly your greatest problem is finding a place to put the Box. The dual drive unit, which Iomega has managed to shrink from the original size, measures 6 1/2 inches high, 12 1/2 inches wide, and 15 inches deep, and weighs about 30 pounds. The documentation does not suggest placing the Box on top of or underneath your system unit. Thus, the unit will occupy some precious desktop or table space near the system unit.

The Box uses a separate controller card (called the *host adapter board*), which you install in a vacant expansion slot. The board can fit into a half-slot, like the slots closest to the power supply on a PC XT. When the board is installed, a single cable connects the Box to the computer. After you connect this cable and the AC power cable, the unit is ready for use.

If you are using the Box as a substitute for a hard disk, you'll probably want the adapter board that allows you to boot DOS from the Box. This adapter is several hundred dollars more expensive than the nonbootable adapter. If you purchase the nonbootable adapter (which was the only adapter available when the unit was introduced), you will need to start DOS from a floppy diskette.

Logical Installation

In order to run, the Bernoulli Box requires a system driver program installed in your CONFIG.SYS file (see Chapter 4). A provided diskette has the Bernoulli INSTALL program, which modifies or creates a CONFIG.SYS file and copies the essential files to the hard disk.

The INSTALL program copies the following command lines to your CONFIG.SYS file or creates a CONFIG.SYS file with these two command lines:

DEVICE = IDRIVE.SYS
DEVICE = ANSI.SYS

IDRIVE.SYS is the DOS device driver for the Bernoulli Box. INSTALL copies this file and the Iomega utility programs to the root directory of the hard disk. The ANSI.SYS file is the DOS screen-keyboard driver discussed in Chapter 4. Two additional programs on the Iomega diskette are IBACKUP and IRESTORE, the Iomega equivalents to DOS's BACKUP and RESTORE. If you want backup and restore capabilities, you need to copy these files to your hard disk.

To start the Box, you must restart DOS. (Press Ctrl-Alt-Del or turn the power off and on again.) After initial parts of DOS load into the computer, you'll see

```
IOMEGA add-on driver version x.y
```

This message appears briefly before the AUTOEXEC.BAT file executes. (The `x.y` is replaced with the current version number of the device driver.) With the device driver, DOS "sees" the Bernoulli drives and assigns each drive the next highest disk drive name. For a system with one hard disk, the Bernoulli drives are D and E. For computers without a hard disk, DOS assigns the Bernoulli drives the names C and D. The Box is now available for use, and logical installation is complete.

The only task left is to format the cartridges. The cartridges can be partitioned, but partitioning is necessary only when more than one operating system will be using the cartridge. The main utility program, IOMEGA, is used to format and partition each cartridge. You can format a 10-megabyte cartridge in about one minute.

IOMEGA, the INSTALL program, several other utility programs, and the Iomega replacements for certain DOS utilities are on the provided diskette. Most of the Iomega utilities can be run as a command-line-driven program (like most DOS software) or can be invoked without options and used as a menu-driven program. Using the menus is easier when you are first starting out.

After you have started the Box, you should copy these program to the DOS or BIN directory on your hard disk or to the *master startup cartridge*, which is a bootable cartridge. You format the master startup cartridge and place on it the DOS operating system, DOS utilities, and the appropriate Iomega utilities.

Backup with the Bernoulli Box

Two types of backup are possible with the Bernoulli Box: cartridge-to-cartridge backup and hard-disk-to-cartridge backup.

IOMEGA is the preferable program for full cartridge-to-cartridge backup. In addition to format and partition options, this program has a COPY option, which performs a track-for-track copy of one cartridge to another. The COPY option is similar to DOS's DISKCOPY, the only difference being that cartridges are copied instead of diskettes. The COPY option is the fastest method for backing up one cartridge onto another (only 2 1/2 minutes for 10M cartridges).

Like DISKCOPY, the COPY option has no capability to back up selected files. You can back up only the entire cartridge on a track-for-track basis. To be selective, you must use a normal file-copy program, like DOS's COPY command or XCOPY program, or other programs mentioned in Part III of this book. The process is slower but offers the capability to back up only parts of a cartridge.

For hard disk backup, Iomega provides the extensive and faster-than-DOS backup and restore programs called IBACKUP and IRESTORE. IBACKUP uses a single full file name, which can include wild cards. If you direct IBACKUP to traverse directories, you can request that the program query you on whether it should back up the contents of each child subdirectory. IBACKUP can select files based on a date (like BACKUP /D) or by archive bit (BACKUP /M). Another option lets you clear the archive bit or leave the archive bit set, allowing greater flexibility for combination cartridge-diskette incremental backups. A feature found in few other programs is the option to "size" but not execute the backup. With this option the program calculates the amount of space the backup will take but does not perform the backup.

Like DOS's BACKUP command, IBACKUP can split backup files on multiple cartridges. Actually, IBACKUP works like DSBackup and places all files from a backup into one large file (see "DSBackup" in this chapter). This technique of packing all backed up files into a single large file is slightly more cartridge-space efficient and fast. Because the files are packed, the traditional dead space at the end of each file is eliminated. The backup process takes approximately 5 minutes for 8.5 megabytes of hard disk files. The time compares favorably with the fastest software backup programs.

The backup file's root name consists of the month, the day, and the number of the backup for that day. The extension is IBK.

For example, the first backup file for July 20, 1986, is labeled 0720-01.IBK.

IRESTORE, the complement to IBACKUP, is used to extract files from the backup. When you are running IRESTORE, two file names are needed: the name of the backup file that holds the files to restore and a full file name (including wild cards) of the files to be restored. You can direct IRESTORE to display each file name as the file is restored and to query you before restoring each file. An APPEND option directs IRESTORE to restore only files that do not already exist on the target disk. Additional options are provided to verify the restored files and to "size" the amount of space required to restore the named files. Also, you can restore the files to any disk drive, including floppy diskettes. You need not restore the files to the original hard disk.

With flexible software options, removable cartridge features, and storage capacity, the Bernoulli Box plays well the roles of hard disk backup device, hard disk storage supplement, or hard disk substitute. The Box is a worthy addition to any computer system. The Box can be the primary storage device or a supplement that can back up faster and more conveniently than the hard disk alone.

Watch that Data: The VCR as a Backup

An idea worth "viewing" is the use of a video cassette recorder (VCR) as a backup device. The first major use of VCRs as backup devices was in 1979 by Corvus Systems, Inc. The Mirror, Corvus' interface, connected a standard (consumer) or industrial VCR to the hard disk system. The benefit is that video cassettes can hold from 50 to 150 megabytes of information and that many people already own a VCR (or can quickly purchase one), making backup less expensive.

Since 1979, Corvus and several other firms, such as Alpha Micro with Videotrax and Kirsch Technologies with Video Memory Manager, have offered units for video backup. The units use an adapter card in an expansion slot. The adapter contains the circuitry necessary to generate a video image of your data. Standard video cables connect the adapter to the VCR, and provided software controls the operation.

The idea of video backup has merit and appeal. The software can provide image, file-by-file, or selective file-by-file backup and usually

offers the standard options of backup by date or archive bit. Although Alpha Micro offers a specially equipped VCR for automated control of the backup, all three adapters can use less expensive consumer units for the task. The only cost, if you already own a VCR, is the adapter with its software and the video cassettes. Because video cassette prices are highly competitive, the cassette's price is a minor expense.

Data is reliably recorded by redundancy. These systems, like many cartridge tape units, record the information several times. If one copy of a block of data is damaged or improperly recorded, one of the remaining copies should be intact. Most units automatically make four copies of the information if no other direction is given. Four copies are usually sufficient. All units allow you to make more copies of the information on the tape; the Videotrax system can be directed to write up to 40 copies of each block of data, which covers virtually anyone's anxiety level.

The disadvantage of VCR backup is speed—or rather the lack thereof. The VCR-based systems are no faster than a low-end tape unit. Restoration is slow also. You have no way to "catalog" the files on the tape. Fast searching of files is difficult to impossible. In most cases, the computer must search the tape sequentially from the start of the recording to the finish.

If, however, you already have a VCR or have multiple uses for the VCR, such as presentations or training, VCR backup is worth investigating. The VCR is a slow but reliable alternative to tape backup units, and the cost of the adapters ($350 to $800 retail) is less than comparable tape units.

Summary

The information described in this chapter gives you several viable options for backing up your hard disk. The software programs FastBack, BackTrack, and DSBackup increase the speed and convenience of making backups. You also may want to consider using a tape drive, the Bernoulli Box, or a VCR as your backup device. Your choice will be determined by your own situation. The important element is that you have a quick and reliable means of backing up your files so that you can avoid loss of valuable data.

Chapter 17

Applications

This chapter provides an overview of how applications packages work (and work best) with a hard disk. Although most current applications developers have written their software to accommodate a hard disk and subdirectory structure, they make these accommodations in different ways.

Throughout this book, I have referred to 1-2-3 and WordStar to illustrate ideas. In this chapter, after a brief explanation of general guidelines, I discuss these and several other popular products. I also show you how you can work effectively with each package on a hard disk.

General Guidelines

The following paragraphs summarize some basic principles that apply to most applications. Following these guidelines will help you get greater benefit from most programs.

The first operations you encounter, of course, are the installation and start-up procedures. The use of batch files and the PATH command usually plays a strong role in installation and start-up.

Installing the Programs

Most applications packages include an automated installation procedure executable by a batch file or by a separate program altogether. This procedure usually takes care of all the steps needed to set up the software. From that point, all you have to do is perhaps reboot your system (if you have modified your CONFIG.SYS file) and start the applications program.

For some applications, however, the installation procedure is thrown completely into your lap. The documentation directs you to create a separate subdirectory on your hard disk, copy certain files from the program diskette to the subdirectory, alter your system's configuration file (CONFIG.SYS), and create a batch file with the commands needed to establish the program's start-up conditions and start the program.

Although this process may sound involved, sometimes I prefer having explicit directions rather than a soup-to-nuts installation program that does everything for me. At least, I know exactly what is happening. Any alteration of my hard disk structure and CONFIG.SYS file as well as the creation of any newly required batch files is done by *me* rather than by some uncontrollable installation program.

Of course, the user can modify the results of any application's installation procedure, and this modification sometimes may be worthwhile. In the discussions of the applications, I include changes I have made to accommodate a package for my own configuration and convenience.

Starting a Program

Generally, you have two techniques for starting an application: from the directory that contains the program files or from the directory that contains the data files or documents you wish to work with. (Because *documents* are the data files of word processors, I use the term *data files* to include documents, too.)

You employ the first technique for programs that must access overlays or help files during operation. The application also may require a special *configuration file* that dictates the start-up conditions, such as where (in which drive/directory) the program finds its data files. For example, Symphony uses a configuration file that has a default path setting for data files.

The second technique usually establishes the directory from which you start as the default data-file directory. This method often

works for applications that load entirely into your machine's RAM, and therefore do not need overlay program files. To execute such a program, you must have its subdirectory (the one containing the program files) already included in a PATH statement, or you must specify the path to that subdirectory when you call up the program.

For example, to start Borland's Reflex, assuming that your data files are in the subdirectory C:\ACCOUNTS and the Reflex program is located in subdirectory C:\REF-PROG, you enter the following commands:

```
CD\ACCOUNTS
C:\REF-PROG\REFLEX
```

The first statement changes the default directory to ACCOUNTS and the second statements calls up the REFLEX program from its location in the REF-PROG directory. Reflex finds your data files in the default directory, ACCOUNTS. If you have previously added to the PATH statement the directory in which the Reflex program is located, you can eliminate the subdirectory reference in the second line.

Fooling an Applications Program

As explained in Chapter 5, you use the DOS PATH command to specify other subdirectories that you want DOS to search for any programs you execute. Thus, if you have given a PATH command that includes the location of your DOS programs, you can execute any DOS program from any directory. PATH works only for main program or batch files, however, not for data or program overlay files.

You can purchase non-DOS utilities that help DOS locate data files and overlay files not residing on the default drive. (FilePath and Dpath20 are popular examples.) These utilities fill in the "hole" in DOS by providing what amounts to *data-path or data-file capability.* Examples of these programs are discussed in Chapter 12.

Another way of "fooling" an applications package is to use the DOS SUBST command (for DOS V3.1 and later). Some applications programs recognize different disk drives but not different subdirectories (WordStar being the most notorious for this). For these programs, SUBST temporarily disguises a subdirectory as a disk drive.

If you have two floppy disk drives labeled A and B, a hard disk drive labeled C, and a RAM disk labeled D, for instance, you can assign the disk drive letter E to a subdirectory of your hard disk.

Then any reference by an applications program to drive E actually goes to the subdirectory (see Chapter 5 for more information).

Organizing Your Hard Disk

A good organizational structure makes navigating around your hard disk easier for you and your applications. Chapter 5 explains ways of setting up an organizational structure suitable for the way you operate. One alternative is a one-level hierarchy which holds all applications and associated data files in separate subdirectories (see fig. 17.1).

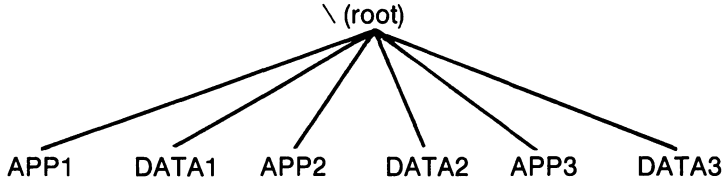


Fig. 17.1. *A one-level hierarchical directory.*

An alternative is to set up a two-level hierarchy. The first level below the root is divided by application; the level below that contains the data-file subdirectories. In the sample hierarchy in figure 17.2, DATA1A and DATA1B represent sets of related data files handled by the same application.

You may want to extend the hierarchy deeper in some cases. For example, you may want to place an application's tutorial files in a subdirectory of the main program's directory (see fig. 17.3). Either way you choose, try to maintain consistency in the hierarchy.

Working with Batch Files

Your computing life is much easier if you take liberal advantage of batch file capabilities. Batch files can help with the housekeeping and establish the necessary start-up conditions: the starting directory,

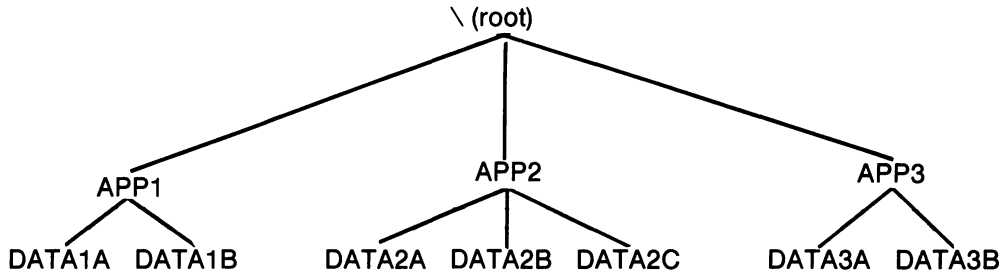


Fig. 17.2. *A two-level hierarchical directory.*

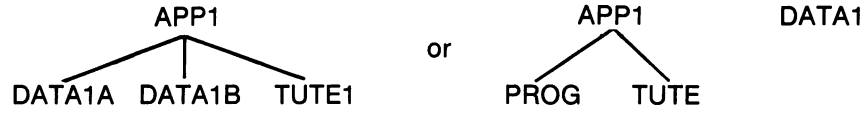


Fig. 17.3. *Variations of the two-level hierarchy.*

configuration or other start-up file, data path, and so on. If you routinely perform some other operation before or after running the application (for example, copy all the files named ACCOUNT1 through ACCOUNT8), you can add those commands to the batch file, too.

Some applications come equipped with their own batch files. There is no reason why you cannot modify the batch files to suit your purposes, "tweaking" (modifying) some command lines and adding others. Batch files can be edited with a text editor (including EDLIN) or a word processor that can work with ASCII text files (often called nondocument mode). You can also use any program that can import text files and write or print to a file.

For example, you can edit a batch file in 1-2-3, but you have to set the left margin to zero and turn off page breaks before you print to a disk file. In fact, you really should not have any formatting at all. With 1-2-3 Release 1A, you can specify only a root name for the file; the program adds the PRN file extension. Once you get back to DOS, you need to rename the file with a BAT extension.

When you have a collection of batch files, you can put them in their own directory, called something like C:\BATCH, and include that directory name in your PATH command. This practice keeps your root directory uncluttered but also keeps the batch files callable from any subdirectory.

You do have to be careful about naming and your PATH conventions. You must consider three elements: the names of your applications package files, the names of your batch files, and the list of directories in the PATH statement. You don't want conflicts among any of these.

Here is an example of confusion that can result from incorrectly placed names in the PATH statement. Suppose that you set up a SYMPHONY.BAT file in the BATCH subdirectory and put the Symphony program files in a subdirectory called SYMPHONY. When you start Symphony, depending on the order of directories listed in the PATH command, DOS might find the program file SYMPHONY.EXE first and execute it without doing all the extras you have included in the SYMPHONY.BAT file.

If you create a BATCH subdirectory to hold all your batch files, you have no need for the SYMPHONY (or any other application called up by a batch file) subdirectory in the PATH statement. As long as DOS can locate the batch file, you can take care of any subdirectory navigation you need in the batch file itself.

The problem is still with you if you happen to start the SYMPHONY batch file directly from the SYMPHONY subdirectory. Because this subdirectory also contains a program file named SYMPHONY.EXE, DOS executes this file without searching for the batch file.

The solution? Change the name of the batch file starting the application. Nothing says the Symphony start-up batch file must be called SYMPHONY.BAT. In fact, to make the procedure convenient for yourself, call the file S.BAT. Your batch file can now combine the conveniences of setting up the application and of requiring only a single key stroke to start. This method also eliminates the confusion of similarly named program files and batch files.

Dealing with Copy Protection

Copy-protected programs sometimes can be installed on a hard disk. With many of these programs, you have the choice of running the program totally from the hard disk or running the program with a floppy diskette in drive A at start-up (the "key disk" requirement).

Of course, installing the program on your hard disk means you won't need the original key diskette every time you want to start the program. The installation process maintains the copy protection visually by creating a few special files hidden in a particular location on the root directory of your hard disk.

With a copy-protected program on your hard disk, you need to be careful when you are backing up, restoring, or performing other structural reorganization of the hard disk. The idea is not to mess up the location or the contents of the special hidden files created during the installation process.

In some backup processes, you can choose not to back up any read-only files. Or, because the restore process is presumably a less frequent and therefore less onerous operation, you can protect yourself (and those hidden files) by remembering not to *restore* any read-only files or to restore only selected files.

If you need to restructure your hard disk by reformatting it or unfragmenting it with the Mace Utilities Condense function or DiskOptimizer, you first must "uninstall" the copy-protected program. When you finish revamping the hard disk, you can reinstall the program. If you forget to uninstall the software, you will probably lose that working copy. (I know. I have done it myself.)

What do you do if your hard disk is damaged? Your first step is to try to uninstall any copy-protected programs. If the portion of the hard disk that contains the read-only files is not damaged, that technique will probably work. Then you can go ahead with the rest of your hard disk recovery routine. For a full explanation of backup and restore procedures, see Chapters 15 and 16.

Applications Packages

This section offers tips for using popular applications software packages with your hard disk. I focus on a handful of the more popular and well-known programs: dBASE III Plus®, PFS:FILE, Q&A, Reflex™ V1.1, 1-2-3 Release 2.0, Symphony Release 1.1, Framework® II V1.0, SuperCalc3® Release 2, SuperCalc4®, WordStar V3.31,

WordStar 2000, Word™, WordPerfect® V4.15, MultiMate Advantage™, and CHART-MASTER® V6.1. I am not trying to give a course in using any of the packages, however, so I do not describe all the features.

You will notice similarities in the way many of the programs work with a hard disk. If your set of applications includes packages not covered here, I hope that you can glean enough general principles from the explanations of these programs to apply the same kinds of concepts to programs not discussed.

dBASE III Plus

dBASE III Plus, the newest sibling of dBASE III and dBASE II, is a relational database product used to store data in a highly structured form. The program can retrieve or extract data according to selected criteria and display the retrieved data in a variety of ways. dBase III Plus has its own powerful programming language that can build a complete, soup-to-nuts application. In addition, the program features a menu-driven interface to perform straightforward database tasks without programming.

Although the program includes multiuser capabilities, I discuss dBASE III only in terms of its single-user operation.

Installation

Although dBASE III Plus V1.0 is copy protected, it can be installed on a hard disk. You can run the program totally from the hard disk or with a floppy diskette in drive A at start-up. As with other copy-protected programs, dBASE III Plus creates hidden files in the root directory of your hard disk. (Ashton-Tate has eliminated copy protection with its update, dBASE III Plus V1.1.)

As I have stated several times, I am not especially enamored with installation processes that fiddle behind the scenes with files on my hard disk. Because most users probably want to use dBASE III Plus without a floppy diskette, however, I concentrate on the hard disk installed version.

If you choose the hard disk installed version, you must be very careful about backing up and restoring your files. You can choose to back up selected files, or you can restore selectively. That is, you can choose not to restore the dBASE III Plus files when you restore others.

If you need to restructure your hard disk by reformatting it or unfragmenting it, you first must "uninstall" dBASE III Plus. This

process safely recovers an installed copy of System disk #1 from the hard disk.

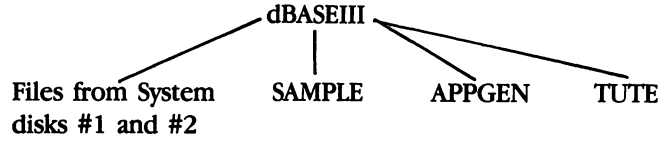
If the uninstall process doesn't work, you have lost that workable copy of dBASE III Plus. But you still have a backup of System disk #1 that can be installed on the hard disk. After you install the program again, you need to get another backup copy.

If you are a registered user (and this factor alone is reason enough to fill out the registration card when you buy the product), Ashton-Tate will replace defective copy-protected system diskettes without charge during the first 90 days and for $20 within three years after that. Why they stop at three years I do not know.

If you are fervently opposed to having those hidden files on your hard disk or if you prefer not having to worry about remembering to uninstall the product, you can skip the hard disk installation procedure. Just copy all the files to a dBASE III subdirectory and start the program with System disk #1 in drive A.

In either installation technique, you should create a separate subdirectory for the program files. If you choose the automatic hard disk installation (copy protection and all), note that the install process copies only the files from System disks #1 and #2. If you want the Sample Programs and Utilities, Applications Generator, and On-disk Tutorial, you must copy those files separately.

Another good idea is to create separate subdirectories of the dBASE III directory to hold the other files, using something like the following structure:



Your system's CONFIG.SYS file must be modified to include appropriate FILES and BUFFERS statements. If you have a 256K-RAM system, you should have statements with at least the following parameter values:

FILES = 20
BUFFERS = 4

For a system with more RAM, the buffers statement should be at least BUFFERS = 20.

A CONFIG.SYS file for the higher RAM systems is already created on the dBASE III Plus System disk #1. The version supplied for the 256K-RAM system is named 256.SYS. If you decide to copy this file, you have to rename it to CONFIG.SYS on your hard disk. If you already have a CONFIG.SYS file, you need to modify it in either case.

dBASE III Plus also has its own configuration file called CONFIG.DB, which is used to customize dBASE III Plus to your own needs. Again, two versions are provided for systems with different RAM sizes.

If you modify your CONFIG.SYS file or create a new one, make sure that you reboot your system before you try to run dBASE III Plus.

Operation

You should add the dBASE III Plus directory to your DOS PATH statement. Then, either directly or from a batch file, change the default directory to the one containing your database files (not the dBASE III program files) and call up the starting program (DBASE). Curiously enough, although dBASE III Plus is a fairly modern program, it doesn't allow you to specify a default *directory* for database files. You can change the default *drive* with the dBASE III Plus SET DRIVE command but not the directory. The start-up process just described changes the default directory before you start the program.

Note that I am making a distinction among the dBASE III Plus program files, the database files, and the program command files you develop (PRG files). The first group actually runs the dBASE III Plus software; the second consists of the files containing your data, and the third is made up of the sets of statements you write (perhaps with the assistance of the Applications Generator).

You can set a path for dBASE III Plus to find your program command files (the third group). This can be done in one of two ways:

1. In the CONFIG.DB file, you add a statement similar to a DOS path statement:

   PATH = C:\DBASEIII\PROGS1;C:\DBASEIII\PROGS2

2. At the dot prompt in dBASE III Plus, you can change the path by typing

   SET PATH TO C:\DBASEIII\PROGS1;C:\DBASEIII\PROGS2

PFS:FILE

PFS:FILE is one of the better known single-file database managers. PFS:FILE is straightforward to use and, as a result, has built up a large following among users who don't need the added sophistication and complexity of a relational database (like dBASE III, R:BASE, Paradox, and others).

Installation

The original PFS:FILE is copy protected. With the later enhanced version, PFS Professional File, Software Publishing Corporation has dropped the copy protection to eliminate the problems associated with backing up hard disks. Because I am working with the copy-protected program, this discussion deals with that version of the program.

The copy protection allows for installation on a hard disk. Before you install the program, you should make a backup copy of the original program diskette. Don't use DOS COPY or DISKCOPY though. PFS:FILE comes with its own BACKUP program. With the program diskette in drive A and a blank formatted diskette ready to use, change to the A prompt, type **BACKUP**, and press Enter.

Depending on how many floppy disk drives your system has, you are prompted to place the blank diskette in drive A or B at the appropriate time. In less than a minute, you have a backup copy. Store the original and insert the copy into drive A.

To install the program on the hard disk, you need to run the SETUP program. First, create a subdirectory to hold the PFS files. This subdirectory can be anywhere you want. Then at the A prompt, type **SETUP** and press Enter. The SETUP menu appears. Press 4 to select `Install program on another disk`. The program prompts you for the drive and directory on which to place the PFS files.

Note that only five installations are allowed. Every time the files are installed on the hard disk, an internal counter is changed on the program diskette. The copy protection takes the form of a single hidden file in the hard disk's root directory. The program checks for the existence of this file whenever you boot up PFS:FILE.

Although the manual indicates that you can set the default (data) drive and directory from SETUP, this capability seems to matter only when you are storing the data on a floppy diskette.

Operation

PFS:FILE can be started in one of two ways. You change to the PFS directory you created and type **FILE**, or you add the PFS directory to your PATH command. Then you can type **FILE** and call the program from anywhere on the hard disk.

You start the program from the default data directory. Whenever you choose an operation that requires a file name, you have several options for your entry in the `Directory or filename` field (see fig. 17.4). These options and their resulting displays are

1. Nothing: Displays all files and subdirectories in the default drive or directory
2. A drive letter: Displays all files and subdirectories in that drive's default directory
3. A drive and subdirectory: Displays all files and subdirectories in that path
4. A full filespec—a drive, subdirectory, and file name: PFS:FILE accesses the file specified

As you can see, you have complete flexibility with regard to subdirectories.

```
                    PFS:FILE MAIN MENU

1.  Design file        5.  Print
2.  Add forms          6.  Remove
3.  Copy               E.  Exit
4.  Search/Update

    Selection:
    Directory or filename: c:\docs

          Copyright 1984 Software Publishing Corporation

F1-Help                                          ↵ Continue
```

Fig. 17.4. *Options menu of PFS:FILE.*

Q&A

Q&A claims to have the same basic user interface as the PFS product line, but with more features and at a better price. Q&A includes a file manager, a report writer, a word processor, and a feature called the Intelligent Assistant. With the Intelligent Assistant, you retrieve and alter information in a database by using ordinary English words and phrases.

Installation

The six diskettes in the Q&A package include four diskettes with program files and two diskettes with sample files and a tutorial. Installation on a hard disk is a matter of creating a subdirectory (called something like QA) and copying all the files into the subdirectory.

The manual stresses that you must copy the diskettes *in order*. Otherwise, the program may not install properly. So copy the diskettes in the order in which they are numbered—#1 through #6. Why? All I know is that several files are found on more than one diskette, probably to accommodate use on a floppy disk system. One of these files is a different size on diskette #4 than on diskette #2. Apparently, you need the file on diskette #4 to overwrite the other one in order for Q&A to operate correctly.

You can dispense with the sample files and tutorial files after you learn the program. Erase them from your hard disk if you need the space.

Q&A does not require a CONFIG.SYS file. If you already have a CONFIG.SYS file, however, you should make sure that it includes lines with the following minimum parameters:

FILES = 10
BUFFERS = 2

The only remaining part of the installation process is configuring the software to recognize your printer. That procedure is done in the UTILITIES module itself.

Operation

You start Q&A by moving to the subdirectory containing the program files and typing **QA** (no ampersand). When you are booting up, the subdirectory from which you start is the default directory for data (both database files and word-processing documents). At any

time you can change the default by going to the UTILITIES MENU and selecting `Set default directories` (see fig. 17.5). You can set different defaults for database files (used by the FILE module) and word-processing documents (used by WRITE module). The UTILITIES MENU is also where you indicate your printer selection.

```
                    UTILITIES MENU

I - Import data          D - DOS file facilities
E - Export data          S - Set default directories
P - Install printer

                    Selection:

Esc-Main Menu        F1-Description of choices        ↵ Continue
```

Fig. 17.5. *Q&A UTILITIES menu.*

Another UTILITIES option is *DOS file facilities.* With this function, you list files, and rename, delete, or copy a file in the default or any other subdirectory.

If you need to locate a file in a large subdirectory listing, you can type the first few characters followed by two periods and let Q&A find the file for you.

When you need a file in FILE, REPORT, WRITE, or UTILITIES, you can type the file name with or without a subdirectory. If you completely omit the file name or type just a subdirectory name, a list of files in the default or selected subdirectory is displayed to help you pick the file you want.

When you create a new data file, do not give the name an extension; Q&A adds its own. For word-processing (WRITE) documents, on the other hand, you can select your own file extension for convenience and categorization needs.

Reflex Version 1.1

Borland International, the firm that helped popularize the Pascal language by making available a $50 version, took over Analytica and promptly dropped the price of Analytica's product, Reflex, from $495 to $99, thereby attracting significantly more attention for the product. (The price has since crept back up to $149.)

Reflex is a single-file database system that offers five different ways of looking at your data: the LIST view (in tabular list form); the FORM view (like a data entry form), the GRAPH view (graphical representation), the REPORT view (arranged in report form), and the CROSSTAB view (allowing you to "cross reference" one variable against another).

Installation

In keeping with Borland's general approach to software, Reflex is not copy protected and is easy to install. You simply create a Reflex subdirectory and copy the files from the three diskettes.

The only configuration you may need to do involves the graphics display adapter (Reflex runs only in graphics mode). If you don't have an IBM Color Graphics Adapter (or an adapter that emulates the CGA), you must run a configuration program called RFLXINST. It gives you a small selection of graphics adapter cards. Printer installation is a matter of choosing a printer port from the Global Settings option in the Print/File menu.

Operation

Reflex comes in two parts: the main database program itself (REFLEX) and the Report and Utilities portion (REFLEX2). You can start each in one of two ways: switch to the Reflex subdirectory and type the name of the program or add the Reflex subdirectory to your system's PATH command so that you can enter the name of the program from any subdirectory.

The subdirectory from which you start becomes the default directory. You can change that by choosing the Global Settings option from the Print/File menu in the main program (see fig. 17.6). When you are retrieving, erasing, or saving a file, you can override the default by entering a different drive or directory name. In the Report and Utilities programs, you can similarly set the default directory, independent of the one established in the main program. The Global Settings page can be accessed from any of the main menu's program options.

```
C:\
Views   Edit   Print/File   Records   Search   Global Settings

                    GLOBAL SETTINGS

Status:  Memory Available:  226/226K Working,   0/0K Expanded
         Records:  0 in Filter,   0 Total in Database

System Directory  C:\
Recalculation:   ☐ Manual  ☒ Automatic
Printer Connection:  Serial ☐ 1   ☐ 2   Parallel ☒ 1   ☐ 2
Printer Has Auto-LineFeed:  ☒ No   ☐ Yes

                  [Proceed]   [Cancel]
```

Fig. 17.6 *Print/File/Global Settings Menu from Reflex.*

In general, when you need to make a choice (files, field names, and so on), you press Reflex's Choices key (F10), and the program displays a list of appropriate entries from which to choose. For example, if you call up the Retrieve File option of the Print/File menu, a window pops on the screen. The default drive or directory is displayed. Below the directory specification is a space for entering a file name (see fig. 17.7). With the cursor pointing to the Name field, you press F10 to get a list of Reflex database files located on the default drive or directory.

```
Views   Edit   Print/File   Records   Search   File

              RETRIEVE REFLEX DATABASE

Directory  C:\
Name

            [Proceed]   [Cancel]
```

Fig. 17.7. *Entering a file name for retrieving in Reflex.*

Reflex can use any expanded memory card supporting the Lotus/Intel/Microsoft expanded memory specification (EMS). Because Reflex loads a database entirely into RAM, expanded memory vastly increases the working size of the database.

1-2-3 Releases 2.0 and 2.01

1-2-3 is the integrated spreadsheet product that took the PC world by storm when the program first came out. 1-2-3 has become a real workhorse in corporate America; the program's tremendous installed-user base makes 1-2-3 a tough monument to topple.

1-2-3 includes three pieces: spreadsheet, graphics, and database. More specifically, all your data is stored in the spreadsheet. The database portion has a single file orientation and is structured within the spreadsheet as well. The graphics are created from the data contained in the spreadsheet.

Installation

To install 1-2-3, you make a separate directory. Then copy into this directory all the files from the System disk, Utility disk, PrintGraph disk, Install Library disk, and View disk. The Backup System disk is a backup of the copy-protected System disk.

To complete the installation, you run a program called INSTALL, which configures the software to your hardware. Move to the 1-2-3 subdirectory and type **INSTALL**. You choose `First Time Installation` from INSTALL's main menu. The configuration process for this version of 1-2-3 is much improved over that for the earlier Release 1A. The Release 2.0 installation program leads you by the hand through the configuration process rather than throwing you to the DOS wolves.

The program asks you questions about your screen display, text printer, and graphics printer and then collects the appropriate software from the Install Library to create a *driver set*, which is used any time you run 1-2-3.

After the driver set is created, you should be able to start 1-2-3. Because of the program's copy protection, however, you ordinarily need to have the System disk in drive A when you boot up. With Release 1A, the installation process ended here. With the current releases, you have another choice.

Hard disk owners can choose to continue the installation process in order to use 1-2-3 on the hard disk *without* a key diskette in drive A. Lotus has included on the PrintGraph disk two programs to help

you do this: COPYHARD and COPYON. COPYHARD is for use on an AT or for systems where the hard disk is not drive C. COPYON is for non-AT systems where the hard disk is drive C.

The instructions for running both programs are given in a separate booklet titled "How To Start 1-2-3 Directly from a Hard Disk." You also need the original PrintGraph and System disks because the installation procedure not only places some protection information on your hard disk but also removes some information from the diskettes so as to allow you precisely *one* hard disk installation.

You place the PrintGraph disk in drive A (drive B if you are working on an AT with a high-density A drive and a low-density B drive). Switch to the drive containing the disk by typing the drive letter, a colon, and pressing the Enter key. Then you type the name of the appropriate program (COPYON or COPYHARD) and follow the directions shown on the screen. The program takes the relevant information off the PrintGraph disk and prompts you to insert the System disk. Again, the program takes the relevant information. After the process is finished, you should be able to start 1-2-3 without the System disk in drive A.

Before you attempt the complete installation on the hard disk, however, make sure that you first go through the normal installation procedure, using the INSTALL program as already described.

Because the hard disk installation process puts hidden files on your hard disk, you must be careful when backing up, restoring, or defragmenting your files. In particular, you must *not* restore those hidden files; restoration can damage the delicate nature of the protection process. For example, if you use the DOS RESTORE command, make sure to specify the */P* option, which asks you whether you want to restore read-only files. Answer N (No) to that question. You can also get into trouble if you run a program like the Mace Utilities Condense function or DiskOptimizer. Because the copy protection often depends on the exact location of hidden files, any program that moves them around on the hard disk may botch up the protection and prevent 1-2-3 from booting up.

After installing 1-2-3, you may want to erase the Install Library file (INSTALL.LBR) from the hard disk directory. You won't need the Install Library file unless your equipment changes and you have to reconfigure the software. For that matter, you can remove all three INSTALL program files. Don't erase any of the files from the floppy diskettes, though; you may need them in the future.

Installation of Release 2.01

One improvement in the so-called "maintenance" update, release 2.01, is the integration of the hard disk installation into the INSTALL process. Thus, the separate COPYHARD or COPYON step described earlier is unnecessary. In fact, after you copy all the System, Utility, PrintGraph, Install Library, and View disk files into a hard disk subdirectory, and run the INSTALL program from the hard disk, the INSTALL program will ask

```
Do you have a hard disk?
```

right off the bat. If you answer **Yes**, the program will ask you if you want to run 1-2-3 from the hard disk. A Yes response will cause the program to guide you immediately through the copy protection process.

(If you are using MS-DOS V2.11, the INSTALL program *cannot* be used to place the copy protection on the hard disk of some IBM-compatible computers. In that case, you will have to run the COPYON procedure.)

Operation

To start 1-2-3, move to the subdirectory containing the 1-2-3 files and type **123** or **ACCESS**. Typing **ACCESS** gives you convenient access to 1-2-3 and its utility programs: PrintGraph, View (the on-line tutorial), Install, and Translate. If 95 percent of your usage is of 1-2-3 itself, you can skip the extra front-end ACCESS menu and start the program directly by typing **123**.

After you start 1-2-3, you will probably want to change the default directory. To reset the directory permanently, you go into the Global default section by typing

/WGDD

These command letters stand for **W**orksheet **G**lobal **D**efault **D**irectory. The slash (/) calls the menu. At the top of the screen, 1-2-3 displays the `Directory at startup` (see fig. 17.8). You type **U** to **U**pdate 1-2-3's configuration file with the new setting.

To change the directory temporarily (for the current session only), you type **/FD** for **F**ile **D**irectory. 1-2-3 displays the current default directory (see fig. 17.9). You can modify the directory setting at this point. When first working with 1-2-3, beginning users can be fooled by the command /File Directory. This command does not list a directory of files.

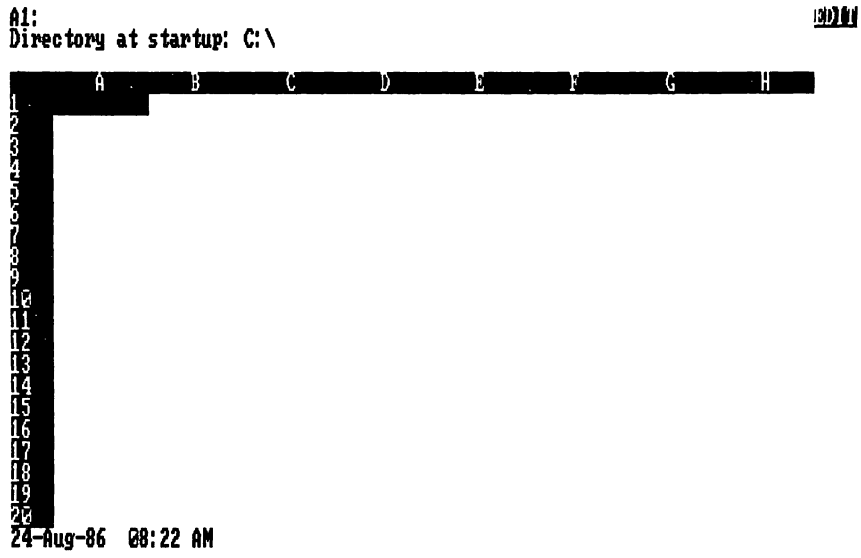


Fig. 17.8. *1-2-3 display of the startup directory.*

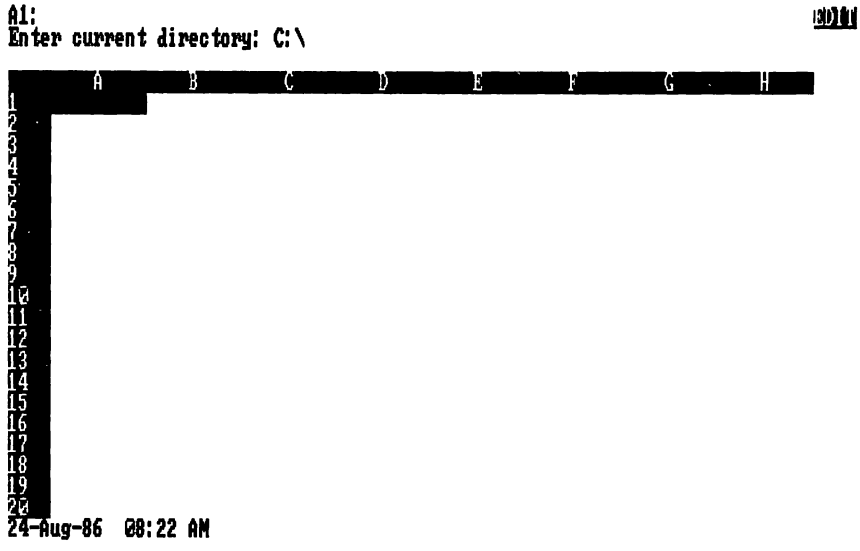


Fig. 17.9. *Changing the current directory in 1-2-3.*

When you are using the File command menu to retrieve, save, combine, extract, erase, or import a file, or to list a set of files, you can temporarily (for that command) override the default drive or directory. The same goes for the Print File menu (when you wish to print to an ASCII text file).

A new feature with Release 2 is the System command, which temporarily puts you at what looks like the DOS level so that you can execute DOS commands and other programs. Actually, you are in a sort of DOS "window" and can return to 1-2-3 exactly where you left off by typing **EXIT** at the DOS prompt. The System command is a replacement for Release 1A's File Manager and Disk Manager commands in the Access menu.

1-2-3 Release 2 can use up to four megabytes of expanded memory conforming to the Lotus/Intel/Microsoft expanded memory specification. Bear in mind, though, that you can fill up conventional memory and not be able to use any more of the expanded memory. 1-2-3 always needs conventional memory to store certain information and to keep track of what is in expanded memory. Therefore, you can exhaust conventional memory and still have plenty of expanded memory left, but not be able to use it. You can check on the status of both memory capacities by executing the **/WS** (Worksheet Status) command. 1-2-3 keeps track separately of the status of each memory.

Symphony Releases 1.1 and 1.2

Symphony was Lotus' followup to 1-2-3 and builds on and expands the same basic structure. Symphony could be called the "4-5-6" to 1-2-3. In addition to the spreadsheet, graphics, and database environments, Symphony includes word processing, communications, and an expanded macro language. Lotus has worked to make Symphony and 1-2-3 even more compatible than in their earlier releases, and that effort shows up in the similarities between the two products.

Installation

To install the program, you create a separate Symphony subdirectory and copy to it all the files from the Symphony diskettes. Next, you switch to that subdirectory and type **INSTALL** to start the installation (configuration) program. From the main menu, you choose the First-Time installation selection. The program asks you questions about your screen display, text printer, graphics printer, and communications equipment. At the conclusion, the program

creates a driver set that includes the appropriate software to control your hardware.

With Release 1.1, you cannot completely install Symphony on a hard disk. You still need the key floppy diskette (Program disk) in drive A every time you start the program. Release 1.2 *does* allow for complete installation onto a hard disk while still retaining copy protection. Like 1-2-3, the installation program asks you if you have a hard disk and want to run Symphony from it. The program then guides you through the copy protection process.

Operation

As with 1-2-3, you can start Symphony in one of two ways. You can type **SYMPHONY** or **ACCESS**. ACCESS displays the front-end menu through which you call up Symphony or any of its utility programs (Install, Tutorial, PrintGraph, or Translate).

Symphony has complete flexibility with regard to subdirectories. The start-up default directory is stored in the Configuration settings sheet. To change the default, you press F9 (the SERVICES key), type **CF** for Configuration File, and modify the drive or directory (see fig. 17.10). You can press either ESCAPE to erase completely the current directory setting or backspace to erase the setting character by character. Type **U** to Update the configuration file if you want the default changed permanently.

```
Initial current directory: C:\SYMPH11

File: C:\SYMPH11                    Document                    Window
Printer                               Tab interval:   5           Type: SHEET
  Type:      Parallel 1               Justification:  l           Name:
  Auto-LF:   No                       Spacing:        1             MAIN
  Wait:      No                       Left margin:    1         Help: Removable
  Margins                             Right margin:             Auto-Worksheet:
    Left:  4      Top:      2         Blanks visible: No
    Right: 76     Bottom:   2         CRs visible:    Yes       Clock on Screen:
  Page-Length: 66                     Auto-Justify:   Yes         Standard
  Init-String:                                                  File-Translation:
  Name: Epson FX, RX, and JX series; LQ-1500                  IBM or COMPAQ
Communications name: C:\SYMPH10\HISPEED.CCF
                                                              Configuration Settings
```

Fig. 17.10. *Modifying the drive or directory in Symphony.*

To change the default for the current session only, you press F9 and type **FD** (SERVICES File Directory). Then you modify the drive or directory name that is displayed. You can modify the default drive or directory any time you use a file-oriented command: File Save,

Retrieve, Combine, Xtract, Erase, List, Table, Import; Print to a file; File-transfer under communications; Image-save under graphics; specifying an automatically executing worksheet or a communications default settings file. You can group your working files into appropriate subdirectories and access any of them directly from Symphony.

Like 1-2-3, Symphony supports the Lotus/Intel/Microsoft expanded memory specification—up to 4 megabytes. Symphony also provides the window to DOS. You must first "attach" DOS as an add-in application by pressing F9 and typing **AA** (SERVICES Application Attach). Symphony searches its subdirectory for any add-in applications, one of which is DOS.APP. You highlight that choice and then invoke it by pressing **I** for Invoke.

Attaching an add-in also adds it to the basic SERVICES menu. As long as the DOS application remains "attached," you can also get to DOS by pressing F9 and typing **D** for SERVICES DOS.

Once in the DOS window, you can run any DOS commands or other programs as long as you have enough RAM. (Symphony stays resident in memory.) This feature is particularly handy for copying files or formatting a new diskette without losing your place in Symphony. This, however, is *not* the time or place to run a large program that could disturb Symphony's RAM residency. This warning also applies to RAM-resident programs which might prevent Symphony from reloading. When you are finished with your DOS operations, you pop back to Symphony by typing **EXIT**. You are right back where you were when you left.

Framework II Version 1.0

Framework II is the second version of the program from Ashton-Tate, the company that also brings you dBASE III and MultiMate. Framework II is another in the breed of *integrated* programs and is often compared with Symphony. The program includes a spreadsheet, word processing, outlining, graphics, data management, and communications. Framework II also has a built-in programming language to let you create custom applications.

Installation

The installation procedure of Framework II V1.0 bears a distinct resemblance to the installation of dBASE III Plus V1.0. This resemblance is no accident because the programs use the same kind of copy protection (SUPERLoK). When you install Framework II on your hard disk, the program tucks away in your root directory three special hidden files. You must leave these files exactly where they are

and the way they are. (For more detail, see the section on dBASE III Plus. Also, note that Ashton-Tate has recently released non-copy-protected versions of Framework II and dBASE III Plus. The discussion of the copy-protected version is presented because of the many V1.0 users who may not upgrade to V1.1.)

Apply the same caution as for dBASE III Plus: don't touch those hidden files during backup or restoration. One way is not to back up your root directory. If not backing up the root directory is too much bother, then back up or restore selectively. Either don't back up the hidden files or be particularly careful that you do not restore them. When you back up to reformat the hard disk or unfragment all your files, uninstall Framework II first.

Installation is a simple process from the user's point of view. Put the Setup diskette into drive A, change to the A prompt, and type **SETUP**. Framework II is entirely menu driven and the first menu choice is SET UP FRAMEWORK II FOR THE FIRST TIME. That's your choice at this point. You use the other option to modify defaults and to uninstall or reinstall Framework II on the hard disk (see fig. 17.11).

```
Welcome to Framework II's "Setup" program.  Choose one of the following:

    (1)  SET UP FRAMEWORK II FOR THE FIRST TIME.

         Start here.  This part of Setup prepares Framework II to
         use your disk drives, screen and one printer.  After you
         become somewhat familiar with Framework II, run Setup
         again and choose option (2).

    (2)  ALL OTHER USES OF THE SETUP PROGRAM.

         You can customize Framework II to your particular
         hardware and personal style in many different ways.  This
         part of Setup provides easy access to all these options,
         including hard disk installation and uninstallation.

    (3)  EXIT FROM THE SETUP PROGRAM TO DOS.

==============================================================================
 PRESS A NUMBER to indicate your choice, using the top row of the keyboard.
        See printed documentation for trademark acknowledgments.
```

Fig. 17.11. *Framework's Setup menu.*

After you select your first-time installation choice and indicate that you have a hard disk, the Setup program creates the subdirectory in which you want Framework II installed (if the directory doesn't already exist) and prompts you to insert the diskettes. Before the

program creates the hidden files in your root directory, the prompt reminds you that you have precisely `0001 installs` allowed. Don't be surprised if the installation doesn't work the first time; I had to try a couple of times before the program worked successfully.

After Framework II is installed on your hard disk, you do not need the diskettes. That is, you do not need them to run Framework II. You should store them in a safe place in case anything happens to the copies on the hard disk. If you ever want to uninstall the program, you need the original System #1 diskette, onto which the hidden files are restored. Ashton-Tate does provide a backup copy of the System #1 diskette in case your hard disk crashes before you have uninstalled the program.

If you uninstall the program to move to another machine, note that all the unprotected program files of Framework II still remain on the original hard disk. After you uninstall the product, you can remove those files from the hard disk with the DOS ERASE or DELETE command.

The Setup program also takes care of configuring the software to your hardware by asking questions about the kind of display and printer you have. After the installation is finished, you will want to customize further by selecting ALL OTHER USES OF THE SETUP PROGRAM.

Hard disk users in particular need to remember that the uninstall procedure is located under the Configuration option. For now, though, you probably want to change the default drive. The Setup program sets this default to the highest drive letter in your configuration. For instance, I have two floppy disk drives (A and B), the hard disk (C), two Bernoulli drives (D and E), and two phony drives created by the DOS SUBST command (F and G). By going into the `Secondary Hardware` section, I can indicate that I want the C drive to be the default drive.

Make sure also that you have a CONFIG.SYS file in your root directory and that the file has a line specifying at least

FILES = 10

If you modify or create your CONFIG.SYS file, reboot your system before you try to use Framework II.

Framework II also can be configured to use *extended memory*, the term Framework II uses for expanded memory, a RAM disk, or a hard disk. In other words, Framework attempts to keep active in RAM any frames you retrieve or create. With several frames brought into RAM, you may soon run out of usable memory. To make room for new frames, you can configure the program to "swap out" the old

frames to expanded memory, a RAM disk, or a hard disk. You specify this feature under the Use of Memory option of the Change Configuration menu of the Setup program.

Operation

To start Framework II, you move to the subdirectory the program is in and type **FW** or add that subdirectory to the PATH command so that you can invoke FW from any place on the hard disk. The default directory is the subdirectory you start in. The default drive and subdirectory can be changed easily within Framework II by positioning the cursor on the drive or directory name and pressing Ctrl-Enter.

The default takes on less significance, however, because you can move easily (in Framework lingo, navigate) between drives and directories. In fact, the navigation is similar to "pointing and shooting" in some of the hard disk managers discussed in Chapter 13. Within a File Cabinet (disk drive), the Down Level (minus) and Up Level (Plus) keys move the cursor through the subdirectory and file hierarchy. These same keys are used to move down and up through the frames hierarchy.

If you need to find a file in a huge subdirectory, you can use Framework's SEARCH command. The SEARCH command is used in any frame to search for a particular combination of characters, and you can also use the command to search for a file. Type the first few characters of the file name and press Enter. You also can use wild-card characters in a file name.

You can change the name (label) of a frame that is active on your screen. A label can be much longer than the eight characters of a file name; Framework II simply takes the first eight legal characters and saves the frame under that root name with an FW2 file extension. If you annotate your frames with useful long labels, however, you may find that feature to be a minor problem because of Framework's rules for storing file names. For instance, suppose that you have a frame labeled

MEMO TO DIDI BONNER CONCERNING BEHAVIORAL CHANGES

Framework stores that frame under the file name MEMO_TO_.FW2. The program replaces any illegal characters—in this case spaces—with underline characters.

This type of file name is not very informative. You can better control the file names by using a semicolon to separate the storable file name from the rest of the label. Suppose that you label the frame

DBMEMO; To Didi Bonner concerning behavioral changes

Framework II picks up legal characters until it reaches the semicolon or has eight characters. This frame is stored under the name DBMEMO.FW2 (see fig. 17.12).

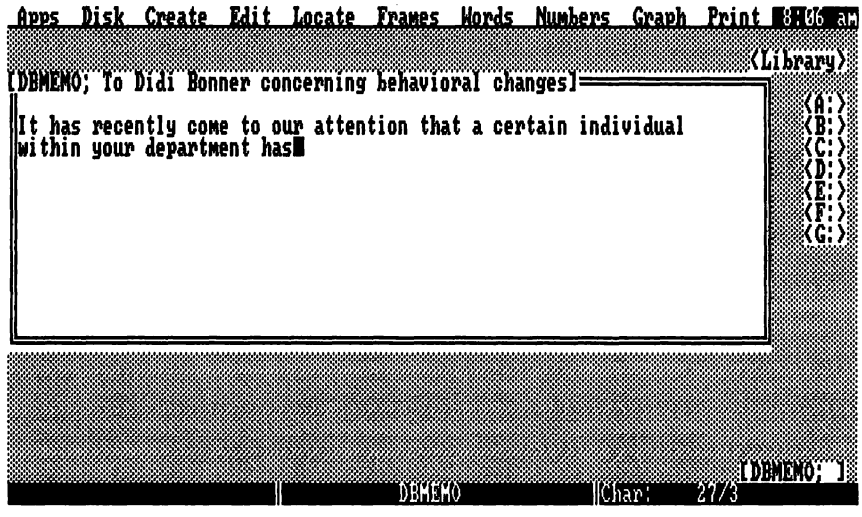


Fig. 17.12. *Storing a file name in Framework II.*

Don't try to include an extension in your frame label. The period (considered by Framework to be an invalid character) is replaced by the underline, and Framework stubbornly files the frame away with an FW2 extension.

A new frame is saved to the default drive and directory unless you specify otherwise. A previously existing frame is saved back to its original location. You can override this location by using the Move or Copy command on the active frame. If a file with the same name is in the drive or directory you pick, Framework II warns you that you are about to overwrite that file.

You can run DOS inside a word frame so that a transcript of the DOS session is saved (up to a user-resetable default maximum of 2,000 characters). The only exception to this is any program requiring a full screen to run. (WordStar and 1-2-3 are the prime examples.) Framework II stops the transcript at the point where one of those programs is started. From a hard disk management perspective, I find that the easier method is to call up a DOS frame,

do my copying, moving, and deleting, and then jump right back to where I left off in Framework. This method seems more efficient than navigating through the directories the Framework way.

Building on the DOS access, Ashton-Tate includes the capability to create a customize list or menu of outside programs you might wish to run. By using the FRED™ programming language, you can use Framework's Add-in Applications menu to set up any program. This feature brings Framework II even closer in concept to the windowing environments discussed in Chapter 14, although the add-in applications cannot run simultaneously. The feature also shares the windowing environments' RAM limitations, which can be partly alleviated by the use of expanded memory, a RAM disk (above the 640K DOS limitation), or the surplus memory of a hard disk.

SuperCalc3, Release 2

SuperCalc, in all its versions, has been around a long time and has been fighting hard against the giant 1-2-3. SuperCalc is a spreadsheet environment that also includes data management and graphics capabilities. The program has always had its fans and fanatics. Although Computer Associates has recently begun marketing SuperCalc4, the popularity of the earlier version warrants its inclusion in this discussion.

Installation

Because SuperCalc3 is not copy protected, installation is simply a matter of creating a subdirectory and copying the files from the two diskettes. Actually, you don't need all the files from the diskettes. The package includes driver programs for both the IBM and Hercules display screens, so you can certainly delete from the hard disk the driver file you don't require for your setup. The package also has a file that contains drivers for various plotters. If you are using only a printer, you can erase this file from your hard disk, too. Generally, you should keep all those files on the diskettes, though. You never know when you may change your equipment and need the other driver sets.

The diskettes also come with a variety of worksheets, which you don't absolutely need but which probably are useful for demonstrating various features of SuperCalc3. Keep the worksheets, look at them, and erase them from the hard disk at some future time if you don't need them.

You take care of the rest of the installation process—configuring the software to your hardware—from within SuperCalc3 itself. To boot up, change to your SuperCalc3 subdirectory and type **SC3**.

To set up the graphics printer configuration, you enter **/GGD** for Global Graphics Device. You follow the screen instructions to select a printer or plotter (see fig. 17.13). Other options may need adjusting too (under Global Graphics Options). You can make these selections or adjustments permanent (until the next time you want to change them, of course) by entering the command **/GKY** for Global Keep Yes.

```
DEVICE SELECTION MENU  [F2 to return to spreadsheet]
Current Device: IBM Graphics Printer (Black & White mode)
Black & White Printers (or Color Printers in Black & White mode):
 IBM Graphics          C.ITOH 8600          HP ThinkJet          QUADJET
 IBM ProPrinter        DIABLO C150          IDS PRISM 132        TI OMNI 850
 IBM 3852 ColorJet     DIABLO P11           IDS PRISM 80         TI OMNI 855
 IBM 5182 Color        DIABLO P32           M.T. MT160           TOSHIBA 1350
 ANADEX DP-6500        EPSON FX100          M.T. MT180
 ANADEX DP-9620        EPSON FX80           MPI Solution
 ANADEX DP-9625        EPSON JX80           NEC PinwriterP2
 ANADEX DP-9725        EPSON LQ1500         NEC PinwriterP3
 Citizen MSP-10        EPSON LX80           OKIDATA 182
 Citizen MSP-15        EPSON MX100          OKIDATA 192
 Citizen MSP-20        EPSON MX80           OKIDATA 193
 Citizen MSP-25        EPSON RX100          OKIDATA 2410
 C.ITOH 1550           EPSON RX80           OKIDATA 82A
 C.ITOH 1550SCEP+NLQ   FUJITSU DPL24 I      OKIDATA 83A
 C.ITOH 24LQ           FUJITSU DPL24 C      OKIDATA 84
 C.ITOH 8510A          GENICOM 3000 SERIES  OKIDATA 92
 C.ITOH 8510SCEP+NLQ   HP LaserJet          OKIDATA 93

Use arrow keys to move cursor.  Press ↵ to select device.
Press <PgDn> for Color Printers.  Press <PgUp> for Plotters.
 24>/Global,Graphics,Device
F1=Help F2=Cancel F9=Plot F10=View
```

Fig. 17.13. *Setting up the graphics printer configuration in SuperCalc3.*

Operation

You can start SuperCalc3 Release 2 either by changing to the appropriate subdirectory and typing **SC3** or by adding the subdirectory to the PATH command so that you can type **SC3** from anywhere on your hard disk.

At start-up, the default directory is the directory containing the SuperCalc3 program files. When you use any file-related command, such as Save, Load, and so on, you have the option of calling up SuperCalc3's DIRECTORY OPTIONS MENU. To call the menu, you press Enter when the program is waiting for a file name. From this menu, you can change the default directory "on the fly." You even

can change to a nonexistent directory, which SuperCalc3 creates for you on the spot (see fig. 17.14).

```
SUPERCALC3 DIRECTORY OPTIONS MENU

Program drive:\directory is
   A:\
Data drive:\directory is
   A:\
Current spreadsheet file is : NONE

DATA DRIVE & DIRECTORY OPTIONS
   C(hange) data drive
   N(ew) to change or make directory
   D(isplay) all filenames
   S(ee) .CAL filenames & graph names

OTHER OPTIONS
   E(nter) filename
   G(raphs) - current spreadsheet

F2 to abort command or return to spreadsheet.


Enter C(hange), N(ew), D(isplay), S(ee), E(nter) or G(raphs)
  ?>/Load,
F1=Help F2=Cancel F9=Plot F10=View
```

Fig. 17.14. *SuperCalc3 DIRECTORY OPTIONS MENU.*

Don't overlook SuperCalc3's built-in path-editing conveniences. To cancel a long path, for example, you type a minus sign and continue with the new path designation. The program even recognizes the dot directory abbreviations (double dot indicating the parent of the current directory, for example). You can display all the files on a directory or just the SuperCalc3 worksheets. At that point, you also can complete the original command with a file name. You can always override the default drive by specifying a path with a file name.

If you want the default directory to stay in effect permanently (until you change it again), you save it just like you save the printer configuration (Global Keep Yes). This rule applies any time that you want a configuration change kept permanently.

Macro files (called eXecute files by SuperCalc) are treated just like any other file with respect to subdirectories. When you call up a macro file to execute, you can jump to the same DIRECTORY OPTIONS MENU described in the preceding paragraphs.

SuperCalc3 can use up to eight megabytes of expanded memory. You have to signal the program that you have expanded memory by means of the GLOBAL OPTIMUM command menu.

In the same menu is the memory Data-space option, which helps you optimize the use of conventional memory (less than 640K) for large spreadsheets. By default, SuperCalc3 loads as much of itself as possible into RAM. This practice speeds performance but also means that less memory is available for your spreadsheets. Ordinarily, this trade-off is the way you want it. For very large spreadsheets that tax the memory capacity of your machine, however, you use the Data-space option. SuperCalc3 brings in only as much of the program as it needs at one time, with the rest of the program coming in as overlays when needed. This technique squeezes down the RAM space that the program itself consumes and leaves more for the spreadsheet data.

SuperCalc4

Although SuperCalc4 is indeed a much enhanced version of SuperCalc3 Release 2, for our purposes the programs are more alike than different in the way they work with a hard disk.

Installation

Other than the fact that the program now comes on three diskettes rather than two, installation is the same as for SuperCalc3.

Operation

You can start SuperCalc4 in either of the same two ways as SuperCalc3, the only difference being that the program itself is called SC4. Probably the biggest difference a SuperCalc3 user notices is that when you press the slash key, the commands are spelled out in front of you. No more guessing what the letters mean.

Similarly, when you invoke any file-related command, such as Save, Load, and so on, you now have the option of calling up SuperCalc4's DIRECTORY OPTIONS MENU by pressing the F3 (Name) key while the program is waiting for a file name. The program displays a mini-file-manager, a list of the files and subdirectories on the hard disk. You can navigate around this list with the cursor (see fig. 17.15). You can change the default directory "on the fly," even to a nonexistent directory, which SuperCalc4 creates for you on the spot.

You still must execute the Global Keep Yes command for a subdirectory change to become a permanent default. An additional convenience has been added for saving or loading a file: the name of the last file saved or loaded pops into the command. If you are

```
FILE LIST
Directory displayed is B:\
Data directory is A:\

*.CAL

<DIRECTORIES>  BLACKJCK.cal  CURRENCY.cal  DEMOS   .cal  FNMAX   .cal
 FNPLOT  .cal  GROWTH  .cal  MACROS  .cal  PAYROLL .cal  SCREENS .cal
 SIMPSON .cal  TABLE   .cal  TAX     .cal  TENBAK  .cal  WINE    .cal

Select <DIRECTORIES> to list subdirectories        96256 bytes free
  7>/Load,
FILE  F2:Edit F4:Update data directory F6:Toggle details
```

Fig. 17.15. *List of files and subdirectories in SuperCalc4.*

working on a spreadsheet and repeatedly saving it, you don't have to type the file name every time.

Macro files (eXecute files) are treated just like any other files with respect to subdirectories. In addition to macro files, you also can store macros in your spreadsheet (as you can in 1-2-3). Both kinds of macros can be written with the extensively enhanced macro language.

SuperCalc4 also recognizes up to eight megabytes of expanded memory and can be adapted to trade off RAM space against speed of operation under the memory Data-space option, just like SuperCalc3 Release 2.

WordStar Version 3.31

Much of what needs to be said about WordStar has already been mentioned in various places throughout this book. Despite the complaint that WordStar is tough to learn, this program is widely used. In fact, WordStar is a real workhorse among word processors. Many people swear by it and many swear *at* it, although a large number of the latter continue to use WordStar. In comparison with some of the more recent entries in the word-processing field,

WordStar's age has begun to show. However, MicroPro has finally released a new version of WordStar, 4.0, in early 1987.

Installation

WordStar is not and never has been copy-protected. After creating a separate subdirectory to hold the files, you copy the files from the WordStar diskette to the subdirectory. Then you change to the WordStar subdirectory and run the installation program, WINSTALL.

In WINSTALL, you configure the software to recognize your printer and change operating defaults associated with the word-processing environment, such as justification, page numbering, insert or overwrite mode, function key definitions, initial help level, system disk drive, and many others.

The system disk drive is an important setting; by default, WordStar looks to the A drive for its overlay and help files. To run the program from a hard disk, you should change the system disk drive to the hard disk. (For some reason, the WINSTALL program asks for a number instead of a letter—the A drive is 1, B drive is 2, C drive is 3, and so on.)

Operation

After reading through other sections of this book, you know that WordStar (through version 3.3) does not recognize subdirectories (WordStar 4.0 does). Indeed, the original program predates the DOS subdirectory structure. WordStar does recognize different *drives*, but that factor doesn't do you much good if you want to store your documents in organized directories on the hard disk. Whenever WordStar is pointed to the hard disk drive, WordStar "sees" only the subdirectory the program starts from.

You have several ways around this problem. The oldest "solution" is to keep a copy of the WordStar program files in each document subdirectory. The user changes to the appropriate document subdirectory and starts one of the copies of the program from there. This method is effective but inefficient. A better alternative is to use one of the auxiliary path programs (for example, FilePath or Dpath20) as described in Chapter 12.

Users with DOS V3.1 or higher can create a phony disk drive by using the DOS SUBST command. I put the SUBST commands in my AUTOEXEC.BAT file as follows:

```
SUBST F: C:\DOCS
SUBST G: C:\BOOK3
```

I also include a statement in my CONFIG.SYS file to allow for drives through the letter G:

LASTDRIVE = G

My batch file for WordStar looks like this:

```
C:
CD\WSTAR
F:
C:WS %1
C:
CD\
```

C:\WSTAR is the location of the WordStar program files. In this way, the default disk drive (in truth, a subdirectory) is set at F, the one I start the program from (see fig. 17.16). If I am working on my book, I can change the "logged disk drive," as WordStar phrases it, to the G drive (actually the subdirectory C:\BOOK3).

```
not editing
                 < < < O P E N I N G   M E N U   > > >
 ---Preliminary  Commands---   | --File  Commands--  | -System  Commands-
L  Change logged disk drive    |                     | R  Run a program
F  File directory     now ON   | P. PRINT a file     | X  EXIT to system
H  Set help level              |                     | -WordStar Options-
 ---Commands to open a file--- | E  RENAME a file    | I  Run TelMerge
   D  Open a  document  file   | O  COPY   a file    | M  Run MailMerge
   N  Open a non-document file | Y  DELETE a file    | S  Run SpellStar

directory of disk F:
 123.APP       APPLICS.MAP   APPLICS.BAK   APPLICS2.DOC  APPLICS2.BAK  CM.APP
 DBASEIII.APP  FIGS          FIGS.DOC      FIGURE10      FIGURE13      FIGURES
 FIGURES4      FIGURES8      FRAME2        FW.APP        MULTI.APP     NC.MNU
 PERFECT.APP   PFILE.APP     PIX           QA.APP        REFLEX.APP    SC3.APP
 SC4.APP       SYM.APP       SYM.DOC       WORD.APP      WS.APP        WS2000.APP

1HELP  2INDENT 3SET LH 4SET RH 5INDLIN 6BLDFCE 7BEGBLK 8ENDBLK 9BEGFIL 10ENDFIL
```

Fig. 17.16. *WordStar directory of files on drive F.*

Note also in the batch file the inclusion of a parameter, %1. This parameter preserves WordStar's option to include a file name in the WS command line so that I can go directly into the file specified at start-up.

WordStar 2000

From a hard disk perspective, the most important difference between WordStar 2000 and WordStar version 3.31 is the capability to recognize subdirectories. This feature is straightforward and no DOS tricks are required.

Installation

Installation is a simple procedure; you can use the provided program or do the installation yourself. The installation program executes the following steps:

1. Creates WS2000, a subdirectory of the root, and another subdirectory under WS2000 called TUTORS
2. Copies the WordStar 2000 files to the first subdirectory and the tutorial files to the second
3. Modifies your CONFIG.SYS file if necessary
4. Creates two batch files in your root directory, one to start the program (WS2.BAT) and one to run the tutorial (TUTORS.BAT)

The program then moves you into the configuration part of the installation program, where you specify the kind of monitor and printer you have.

Although you may want to rearrange some things, you will probably run the automatic installation program (called WS2COPY for hard disk users). The only change the program makes to the CONFIG.SYS file is to increase the FILES line to FILES = 20 if the value is not at least that number already. For some of the program's more powerful operations, WordStar 2000 may require up to 20 files open at once. The manual also suggests that you use DOS buffers (also set through the CONFIG.SYS file) to speed WordStar 2000's operation. For a system with 256K RAM, the manual recommends 15 buffers; above 256K RAM, the manual recommends 20 buffers.

You may want to modify the location and contents of the batch files WS2.BAT and TUTORS.BAT, which are created by the installation program. If you are keeping all your batch files in a separate subdirectory, you can move WordStar's batch files there, too.

Because of the way the provided batch files are written, you do not even need to add the WordStar files subdirectory to your PATH command. If you do, however, you may want to edit the WS2.BAT file to remove the lines that save the current path, set a temporary

path to the WordStar files, and reinstate the earlier path when you exit from WordStar 2000 (see fig. 17.17). If you add the WordStar directory to the PATH command, you then don't need the WS2 batch file to start the program.

```
echo=off
path > C:\ws2path.bat
path C:\ws2000;C:\
cd C:\ws2000
WS2 %1 %2
cd \
ws2path
rem * If the hard disk where WordStar 2000 is located is labeled by a letter
rem   other than "C:", change all occurrences of "C:" to the letter of your
rem   drive.
rem * If you placed WordStar 2000 in a directory other than "ws2000," change
rem   the first two occurrences of "ws2000" to the name of the directory you
rem   selected.
rem * The 2nd line saves the system PATH.  Line 3 creates a new PATH.  The
rem   directory is changed on line 4 (if you do not want the directory changed,
rem   remove this line.)  WordStar 2000 is executed on line 5.  The directory
rem   is changed to root (\) on line 6.  The final command "ws2path" puts the
rem   path back the way it was before running WordStar 2000.

A:\ —>
```

Fig. 17.17. *WordStar 2000 batch file provided with the program.*

As a nice touch, the WordStar 2000 documentation also includes a section on "Freeing Up Space on Your Disk." It tells which files are expendable under certain conditions. These files are divided into the following categories:

1. Needed only during installation
2. Needed only to present information (as a README file)
3. Needed only for the installation of special features (sheet feeder, change function keys or colors)
4. Needed only for WordStar 2000 training
5. Needed only if you are using a particular printer
6. Needed only to access the help screens

You are free to erase any of these files if you know you won't need them.

The documentation also suggests how the use of a RAM disk can speed the program's operation. Because many program or subsidiary files are kept on the hard disk until needed by the main program, putting these files in a RAM disk accelerates the transfer of information. MicroPro also offers a "RAM version" of the program for machines with at least 512K of RAM. This version loads completely

into RAM from the start so it is faster than the standard model with its program overlays. Until June 30, 1986, MicroPro gave this version free to any registered user who asked for it.

WordStar 2000 is not copy protected so the issues of backup and restore are not problems. The last step, if you modified or created a CONFIG.SYS file, is to reboot your system before you use WordStar 2000.

Operation

Many people are satisfied with the installed locations of WordStar 2000's special files (dictionaries, key files, or format files). However, you may want to keep these files in locations other than those specified in the installation process. For example, you may want a separate FORMATS subdirectory to keep together all the format files. WordStar 2000 provides several different ways of telling it where the special files are. One way is by changing the location from within the installation program itself. The option to change the installed location is particularly useful when you work with WordStar 2000 on a local area network.

An important feature is the ease with which WordStar 2000 handles subdirectories of documents. You can specify a full path when asking to edit an existing or new document. You also can change the default directory (yes, I said directory, not just default disk), to which the program automatically goes to access a document (see fig. 17.18). This default stays until you change it again, even when you exit from the program and reenter it. The default directory is only for documents; WordStar 2000 maintains the installed locations for program overlays, format files, key files, and dictionaries.

```
C:\DOCS
                    C H O O S E   A   N A M E
 Type or highlight name.  Press Return.                    ^Get help
                                                           Escape
    Move highlighting with cursor keys.
    Erase errors with Backspace.
    Transfer highlighted letters to answer line with ^T.
                 ^G means hold down Ctrl key and press G.

Change directory or disk drive to?
```

Fig. 17.18. *Changing the default directory in WordStar 2000.*

In further recognition of the subdirectory concept, WordStar 2000 also includes a utility command—MOVE/RENAME—that works across subdirectories.

Word

Microsoft Word is a full-featured word processor of the "WYSIWYG" (What You See Is What You Get) variety. That is, Word has the capabilities to display print attributes (underlining, italics, boldface, and so on) directly on your screen, not as hieroglyphic control characters.

Microsoft has recently started removing copy protection from many of its products, including Word V3.0. That change, of course, also eliminates problems with backup and restoration. Once you install Word on the hard disk, you can store the original diskettes in a safe place. With Word, you don't need to worry about how you back up and restore your hard disk.

Installation

Microsoft provides a program (SETUP) to take care of installation and initial configuration. SETUP copies into the hard disk subdirectory you choose all the program and other files needed to run Word. If the directory doesn't already exist, SETUP creates the directory.

The program also asks you questions about your printer and your mouse (if any) and copies the appropriate files needed to run these peripherals. You are given the option of adding the Learning Word programs (the on-line tutorial). If your hard disk is cramped for space, you can remove the tutorial files after you have learned how to work with Word. SETUP generally doesn't copy more than Word needs, so you really don't have any other extra files to erase from the hard disk (unless you are not interested in using the spelling checker and associated utilities).

Operation

After you put the Word subdirectory in your PATH command, you can run the program from any subdirectory. The first time you run Word from a subdirectory, that subdirectory becomes the default. At any time from within the program, you can change the default directory (or drive), and the change is kept until changed again, even if you exit from the program.

The name of the last document on which you worked also is saved. You can call up the same file the next time you go into Word by typing

WORD/L

The **L** stands for **L**ast document. You also can specify a document name when you call up the program:

WORD \BOOK\WPROCS

In general, Word has a great deal of flexibility with regard to hard disk files and subdirectories. You can temporarily override the default directory by specifying a full path name when loading a document from within the program. By pressing a function key (F5), you can cause Word to execute its List Files function. The program then gives you a list of files matched by file specification, including the wild card characters * and ?, on the default or any directory (see fig. 17.19).

```
ARC.DOC                IBM32.DOC          LW.DOC          NUS903.DOC

TRANSFER LOAD filename: ARC.DOC            read only: Yes(No)

Enter filename or select from list (11714560 bytes free)
Page 1   {}                                Microsoft Word:
```

Fig. 17.19. *List of Word documents.*

If you don't specify a file extension, Word supplies its own to documents (DOC), style sheets (STY), glossaries (GLY), and document dictionaries (CMP). A style sheet is a Word format file,

and a glossary is the program's storehouse for *boilerplates*: frequently used phrases, sentences, or paragraphs that can be brought easily into documents. A document dictionary is just what its name implies: a dictionary for a particular document.

Word gives you a temporary exit door to DOS under the Library Run command. As a result, you don't have to leave the program entirely in order to perform some needed operation (format a diskette or copy a group of files, for example). Ordinarily, this exit door is not meant to run anything large because Word is still sitting in the machine's RAM. Another caution is not to run any RAM-resident program through this DOS door. You could endanger your chance of returning to Word. Included in this category of ram-resident programs are DOS commands ASSIGN, GRAPHICS, MODE, and PRINT and the many non-DOS RAM-resident programs, like SideKick.

You can run Word from a RAM disk. The manual specifically lists the minimum files to be copied to the RAM disk. As with all RAM-disk situations, the best method is to create the RAM disk out of extended memory (RAM above 640K in an IBM AT). That way, you don't tie up the usable working RAM needed by your applications.

WordPerfect Version 4.1

WordPerfect is another full-featured word processor that has recently seen an upsurge of interest and popularity.

Installation

Installation of WordPerfect is a matter of creating a separate subdirectory anywhere convenient in your root structure and copying into the subdirectory the WordPerfect files from all the diskettes except the Printer disk. For organizational efficiency, you create a separate subdirectory for the program's tutorial or practice documents.

Unlike most other word processors, with WordPerfect you select a printer and specify printer characteristics from within the program itself. A separate setup program is used for further customizing WordPerfect to your operating configuration (see the following section).

Operation

You can start the program by changing to the WordPerfect subdirectory and then typing **WP**. Alternately, you can add the WordPerfect subdirectory to the PATH command so that you can type **WP** from any subdirectory that contains (or will contain) your documents. The start-up subdirectory becomes the default document directory. This default can be changed easily while you are working in the program.

After you boot up the program, you select the Printer Control menu, choose your printer from the displayed list, and specify several printer characteristics (see fig. 17.20). You need the Printer diskette for this procedure because the list of available printers is stored on this diskette. After this process, you can stay in WordPerfect and start working.

```
Printer Control

                                        C - Cancel Print Job(s)
1 - Select Print Options                D - Display All Print Jobs
2 - Display Printers and Fonts          G - "Go" (Resume Printing)
3 - Select Printers                     P - Print a Document
                                        R - Rush Print Job
Selection: 0                            S - Stop Printing

Current Job

Job Number: n/a                         Page Number:  n/a
Job Status: n/a                         Current Copy: n/a
Message:    The print queue is empty

Job List

Job  Document          Destination          Forms and Print Options


Additional jobs not shown: 0
```

Fig. 17.20. *WordPerfect Printer Control menu.*

When you get accustomed to WordPerfect, you may want to run the separate SETUP program to change some of the program's operating options. SETUP is run by typing **WP/S**. Of particular interest to hard disk users are the options to set the directories containing the Speller and Thesaurus, if you have placed them in other than the WordPerfect subdirectory, and to set the backup options.

Because an on-line dictionary can be added to, some people prefer to keep it in a subdirectory separate from the operating programs. If you do this, you must tell WordPerfect where the dictionary is by means of the SETUP program.

Many word processors automatically make a backup copy of the file you are editing. In that way, you can always go back to the earlier version even after you save the edited document. Of course, this method takes extra time, and these extra backup copies consume extra space on your hard disk. WordPerfect can go either way; the default setting is not to make backup copies, but you have the option of changing this in the SETUP program.

Another backup option is *timed backup*. To protect against losing a large section of text in the event of a machine or power failure, you can have WordPerfect automatically back up, at selected intervals, the document you are editing. The program saves the document you are editing under the name {WP}BACK.1. If you are taking advantage of WordPerfect's provision to edit two documents on the screen, the program saves the second document under the name {WP}BACK.2.

If your machine "hangs" while you are in WordPerfect or you have a power failure, you can retrieve the timed backup copy ({WP}BACK.1 or {WP}BACK.2) when you get back into the program. When you restart WordPerfect, you see the following message:

```
Old Timed Backup file exists: 1 Rename; 2 Delete
```

Because WordPerfect keeps only one set of timed backup files, the program needs to be told what to do with the existing ones. If you don't need the files, delete them.

Neither automatic nor timed backup is any substitute for a regular routine of backups. WordPerfect and I are really using the word *backup* in two different ways. Timed backup is intended *only* to protect against a machine or power failure. If you exit from the WordPerfect program with no problems, the timed backup files are deleted. You should be saving your file regularly anyway. Personally, I would rather do it myself.

The subdirectory from which you start the program is the default directory. You can easily change to another document subdirectory by calling up the file management option, List Files. WordPerfect displays a list of your files (see fig. 17.21).

If you need to execute another program, you have a temporary DOS "door" through which you can move out, run the program, and pop back to WordPerfect without losing your place. As I have

```
08/24/86  08:51          Directory C:\DOCS\*.*
Document Size:       0                                  Free Disk Space: 11708416

. <CURRENT>    <DIR>                          .. <PARENT>    <DIR>
AACA    .          1  08/24/86 08:49           ARC     .DOC   60168  01/31/86 15:07
CACHE   .       2191  07/27/86 11:30           FUNDAY2 .       1664  05/10/86 23:21
HD      .WRK   22277  07/27/86 08:29           HDISK   .TXT    6272  08/16/86 13:18
IBM32   .DOC   11776  05/26/86 23:21           LU      .DOC    6784  01/01/80 15:07
NUSQ03  .DOC   13034  07/12/85 15:07           OUTL    .WRI    8651  08/21/86 21:03
OUTLNEW .WRI    9353  08/17/86 07:50           PEGG3   .       7936  06/09/86 23:21
PRECUR  .       1627  07/19/86 10:42

1 Retrieve; 2 Delete; 3 Rename; 4 Print; 5 Text In;
6 Look; 7 Change Directory; 8 Copy; 9 Word Search: 0
```

Fig. 17.21. *WordPerfect list of files.*

mentioned with other programs, you must know how much RAM you have left to run anything else. WordPerfect is still sitting in RAM while you are running the other program.

If you have plenty of RAM in your machine (at least 384K), you may want to boot up WordPerfect with the R option:

WP/R

This command loads more of the start-up program into RAM and so speeds operation. The manual states that using WordPerfect with a RAM disk does not buy you much in terms of speed improvement. If you routinely use a RAM disk on your machine, however, you may want to place the Speller and Thesaurus there (making sure to indicate the location changes through the SETUP program). You also can tell WordPerfect to write its overflow files to the RAM disk. (Overflow files are the temporary files that the program creates on the disk when a document gets too large to keep entirely in RAM.)

MultiMate Advantage

The first MultiMate word processing program was modeled after the Wang dedicated workstation word processors. MultiMate was bought by Ashton-Tate, the owners of dBASE and Framework.

Installation

The program has no copy protection. To install MultiMate Advantage, you create a MultiMate subdirectory (suggested name MM) and copy the files from the seven diskettes: Boot, System, Printer Tables, Advanced Utilities, Speller, Thesaurus, and Advanced Conversions. To save space, you are instructed to copy only the printer files you need for your printer. An eighth diskette—the interactive tutorial—can be used separately.

Before you use MultiMate Advantage, make sure that your system has a CONFIG.SYS file and that it contains a FILES statement of at least

FILES = 15

That's all you do for installation. To complete the configuration for your hard disk setup, you need to start the program. If you have modified or created the CONFIG.SYS file, make sure to reboot your system first.

Operation

To start the program, you first go to the subdirectory containing the program files. Enter the command **CD\MM**. You may want to add to your collection of batch files with a MultiMate batch file like the following:

```
CD\MM
MM
CD\
```

The program can be started by one of two commands: MM or WP. MM boots up the menu, from which you can choose the word processor or file conversion and other advanced utilities. You also run other programs from this menu. Choosing the word processor gets you to Advantage's main menu (see fig. 17.22). If you start with WP, you bypass the first menu screen and are taken directly to Advantage's main menu.

As soon as you get to the word processor's main menu, you ought to finish the configuration process by calling the Other Utilities menu and selecting the Edit Drive Defaults option. A fill-in-the-blanks screen pops up; on this screen you indicate to MultiMate the location of various files: the drive that will house the program files (System Drive), the library documents (Library Drive), the speller/dictionary files (Dictionary Drive), and your word-processing documents

MultiMate Advantage*
Professional Word Processor Version 3.50
(C) Copyright 1985
Multimate International Corporation

1) MultiMate Professional Word Processor
2) MultiMate On-File
3) MultiMate Advanced Utilities and Conversions
4) MultiMate On-File Utility

8) Other User Programs
9) Return to DOS

DESIRED FUNCTION:
Enter the number of the function; press RETURN

THE PROXIMITY/MERRIAM-WEBSTER LINGUIBASE (C) Copyright 1983 Proximity Devices Corporation (C) Copyright 1983 Merriam-Webster Inc.	MULTIMATE ON-FILE* (C) COPYRIGHT 1984, 1985 Computer Dynamics Products Inc. All Rights Reserved

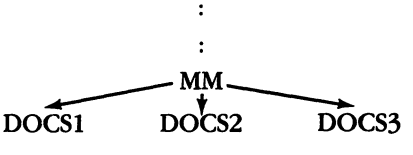
*Trademark of Multimate International Corporation

Fig. 17.22. ***MultiMate Advantage main screen.***

(Document Drive). Library documents are the files that contain frequently used phrases, sentences, and paragraphs, so-called boilerplate documents (see fig. 17.23).

For a hard disk system, the configuration is easy; all the choices probably will be the hard disk drive, usually C, unless you set up a RAM disk to contain the program or speller/dictionary files. Note that the choices are drives, not directories.

MultiMate does recognize subdirectories to a certain extent. You can have different subdirectories for different sets of documents but only in or immediately below the MultiMate subdirectory. In other words, if MM is the program subdirectory, you can create document subdirectories as follows:



```
MultiMate Advantage
Version 3.60

1) Edit an Old Document
2) Create a New Document

3) Print Document Utility
4) Printer Control Utilities
5) Merge Print Utility

6) Document Handling Utilities
7) Other Utilities
8) Spell Check a Document
9) Return to DOS

DESIRED FUNCTION:

Enter the number of the function; press RETURN
Hold down Shift and press F1 for HELP menu
                                                S:1 N:1
```

Fig. 17.23. *MultiMate word processor's main menu.*

Documents can also be stored in the MM subdirectory itself (although I don't particularly like that idea).

Within the DRIVE DEFAULTS MODIFICATION screen, you also can specify a default directory. It is one of the eligible subdirectories (described previously) you enter in this screen. This default directory is searched first for any documents. When you select a command that operates on a document, you can cycle easily through the other valid subdirectories, if you wish.

This subdirectory limitation applies only to document files. The limitation does not apply to file merge and file conversion utilities. For these functions, MultiMate can scan and operate across the entire hard disk.

The Copy and Move commands, Key Procedures (macros), and Library documents cannot move across subdirectories. Therefore, if you use MultiMate's Document Handling Utilities to copy a document, you can copy only within a subdirectory or between disk drives. Move works only between disk drives. (I couldn't fool it by setting up a phony drive using the DOS SUBST command, either.)

CHART-MASTER Version 6.1

Just as a spreadsheet allows you the flexibility to try out different scenarios, a graphics program on a PC lets you experiment with ways of displaying data in graphics form. Although some spreadsheet programs include graphics capabilities, CHART-MASTER is one of the more popular stand-alone programs that help you create presentation-quality graphics.

Installation

Before installing the program, you should create a subdirectory to contain the CHART-MASTER files. Then you must decide *how* you want to install the program.

CHART-MASTER gives you two options for installation on a hard disk; both options maintain the program's copy protection but in different ways. You can choose to install the program entirely on your hard disk so that running CHART-MASTER doesn't require the program diskette in the A drive. On the other hand, if hidden files on your hard disk bother you, you can choose the modified installation process, which copies all the needed files to the hard disk and still requires the CHART-MASTER Diskette 1 in drive A when you are booting up.

The choice is difficult. As I have made clear, I have no love for the hidden file trick because of the complications in backing up and restoring the hard disk. The hidden file is placed in the CHART-MASTER subdirectory. The program's documentation makes the choice even scarier by implying that you get only one installation. Technically, that is correct. But the manual does not point out that CHART-MASTER also has an uninstall program on Diskette 1. If you need to uninstall the program (and still retain its future "installability"), you can.

The complete hard disk installation program is a batch file called INSTALHD.BAT (the uninstall program is DEINSTHD.BAT), and the modified installation program is CREATEHD.BAT. If you go the complete installation route (with INSTALHD), make a note of it so that you don't accidentally mess up that hidden file.

The further configuration of the software to your particular hardware takes place when you boot up the program.

Operation

You start CHART-MASTER by changing to the CHART-MASTER subdirectory and typing **CM**. The first time you boot up the program, it automatically gives you the configuration routine. In configuration, you specify the drive that contains the program and the default drive for data files (graphs) and answer other questions relating to printer type, plotter type, light pen, display options, and so on. CHART-MASTER saves this configuration in a file and displays the opening program menu on the screen (see fig. 17.24). The saved configuration will be used in succeeding sessions and can always be changed.

```
◄◄◄ CHART-MASTER MAIN MENU ►►►

1 CREATE A CHART
2 VERIFY CHART
3 PLOT / PREVIEW CHART
4 EDIT CHART
5 STORE / RETRIEVE / DELETE CHART
6 CHANGE PLOTTING OPTIONS
7 CHANGE / DISPLAY CONFIGURATION
8 RUN DATAGRABBER
9 QUIT MENU

↑↓ Move through choices        ↵ Select

<F1> Help
```

Fig. 17.24. *CHART-MASTER opening program menu.*

Note that you indicate the default *drive* for storage of data files. CHART-MASTER V6.1 does not recognize subdirectories other than the one in which you install the program. When you choose the hard disk drive as the default chart or data directory, all chart files are stored in the CHART-MASTER directory.

Any time you store, retrieve, or delete a chart, or call for a list of charts on a drive, you can override the default drive with a new drive designation. CHART-MASTER also provides the capability to load in a file of data so that you can avoid having to type the data to be

plotted. This DataGrabber feature can read files in special DIF or SYLK formats as well as ASCII text files containing data in a prescribed format. In these cases, too, CHART-MASTER recognizes only drives, not directories. Thus, to "grab" a file from the hard disk, you must make sure to copy the file into the CHART-MASTER subdirectory before you run the program.

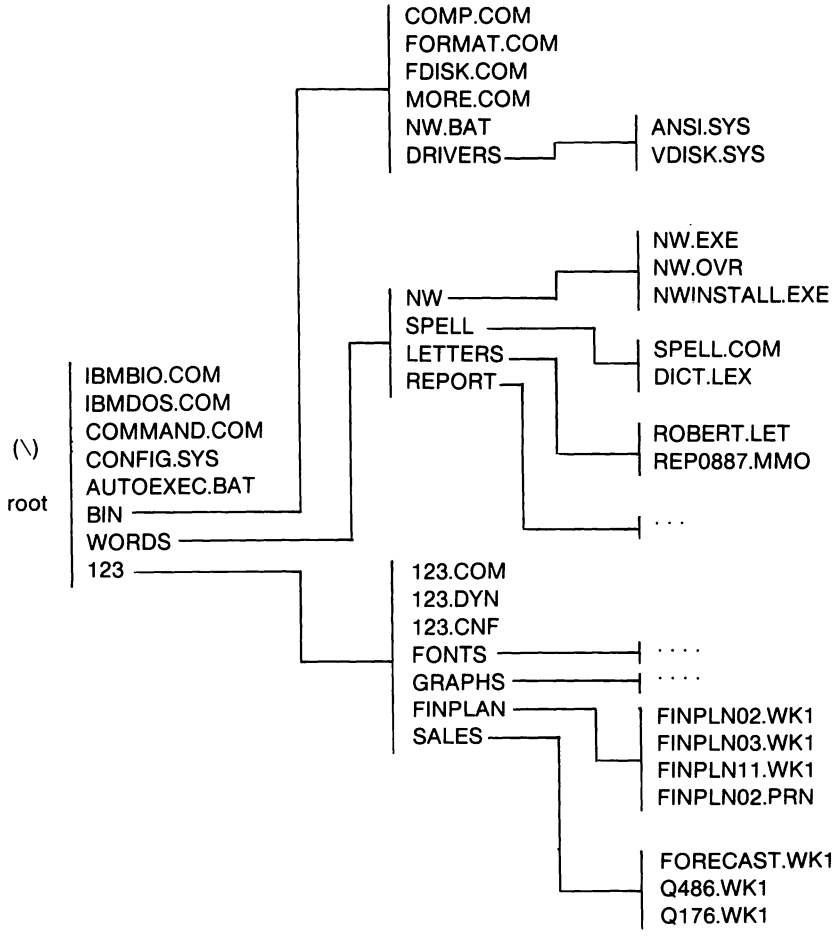
Summary

In this chapter, the following key points were discussed for each program:

- The extent to which certain programs recognize and work with subdirectories
- A review of how to "fool" the applications packages that don't recognize hard disk subdirectories
- Installation of the software, including possible changes to CONFIG.SYS and AUTOEXECT.BAT files
- The two basic ways of starting an applications program: moving to the program subdirectory and calling up the program (usually with some sort of configuration file attached), or moving to the data subdirectory and calling up the program from there
- Ways to organize the hard disk subdirectories for more efficient management of multiple applications
- Tips for using batch files with your applications
- Tips relating to the specific applications

Appendix A

Sample Hierarchical Directory



Appendix B

Resources

Public-Domain and Shareware Programs

Throughout this book, reference is made to public-domain and shareware software. The public-domain and shareware concepts are really unique marketing and distribution channels separate from the normal commercial routes.

Public-domain programs usually refer to computer programs that are freely available. Except for the cost of acquiring the program, no charge is made to the user.

The motivation for an author to place a program in the public domain is usually a combination of generosity and the desire for modest fame. Because no formal charge is made for the program, it quickly winds up in the hands of a diverse collection of users.

By placing the program in the public domain, the program's creator relinquishes both control and responsibility for the program. The public-domain program has no usage restrictions except that the program cannot be sold or resold or, for some programs, cannot be used commercially. The author does not receive any type of royalty. Nor does the author have any marketing or support costs. Public-domain programs are *unsupported:* the onus for learning to use the

program, fixing problems, or improving the program is on the user not the creator.

Because any programmer can place a program in the public domain, these programs have no quality control or formal support. Many public-domain programs have limited to very limited use (read "junk"); however, some public-domain programs are true gems.

The shareware concept grew out of the public-domain idea. *Shareware* programs are freely available to anyone who wishes to try them; however, those who wish to continue using the program have a moral obligation to send to the creator a modest suggested contribution (usually around $10 to $50). Obviously, the creator is trusting the honor system to work. Some additional incentives are often dangled in front of prospective users to get them to pay: more complete documentation, telephone troubleshooting support, free updates if the program is enhanced, and so on. The information relating to contribution and incentives usually is provided in an introductory screen displayed when the program is first run.

Some examples of shareware successes are PC-Talk (a communications program), PC-File (a database manager), and PC-Write (a word processor). PC-File is reputed to have an estimated 250,000 users. The quality of these and other shareware programs is excellent.

How do you obtain public-domain or shareware programs? You can get copies from a current user (the programs are never copy protected), a local or national users group, or from an electronic bulletin board.

Friends, relatives, and acquaintances are potential current users of a product. User groups, a club of people with similar or a wide range of interests in computers, have been organized in many communities. Your local computer store or associates can give you information about these groups.

An electronic bulletin board is a computer equipped with the appropriate hardware and software so that other systems equipped with a modem and the right communications software can fetch files from the bulletin board. Files (the programs) can be downloaded from the bulletin board (sent over the phone lines and stored on the receiving computer's disk drives). Many bulletin boards are operated by user groups or interested individuals. Some bulletin boards are free. Some require membership in the owning club. Some others require a modest annual or onetime fee. In all cases, you pay for the phone call to the bulletin board.

Commercial data services like CompuServe and The Source have sections devoted to particular computer groups (SIGs or Special Interest Groups). These sections contain extensive libraries of downloadable files. The services also encourage users to upload (send) files of their own to share with others. You must formally sign up with CompuServe or The Source, and you will have to pay by the minute for any time you use the system (called *connect-time* or *access* charges).

A Caution and Warning

Although public-domain and shareware programs are considered free or very inexpensive, you may pay in other ways; hence this caution about public-domain and shareware software. Public-domain programs have little or no support. As mentioned, the documentation can range from sketchy to nonexistent. You are responsible for learning how to use the program, changing the program if it does not work the way you wish, and correcting any problems with the program. If the program is complex, the true cost in learning to use the program can be high. Simpler programs will have lower costs.

Some shareware products are well supported by their authors; some have modest support. However, the economics of shareware is that the author does not have unlimited time to support the product. You have more burden for learning the shareware product than for learning commercial products. Part of the cost of a commercial program is the hidden charge for customer support. Shareware, which is offered at a lower price, does not have these charges; and the author cannot afford to hold the users' hands as commercial publishers can.

To paraphrase the axiom on public-domain software quoted earlier, "If you have more money than time, buy commercial. If you have more time than money, public-domain (and shareware) software is wonderful."

A warning comes from a recent problem in the microcomputer users area. Along with the tremendous increase in interest in public-access bulletin boards has come a surge in insidious pranksters and thieves. The bane of the industry is the Trojan horse, along with pirated and hacked programs.

A Trojan horse is a program that claims to do some nifty task but instead pulls some nasty trick on you, like deleting every file on a disk, corrupting the file allocation table, or reformatting the disk. Although rational people find this difficult to believe, some

anonymous "clowns" (the kindest word the editors will let me use in a family book) out there take a perverse delight in wreaking havoc on strangers' computer systems.

Pirated and hacked programs are illegally distributed or illegally disguised and distributed programs. Pirated software is commercial software distributed without payment to the program's publisher. Hacked programs are thinly veiled attempts to remove or alter the copyright notice of commercial software or are distributed versions of "cracked" copy-protected commercial programs. Both forms are illegal and immoral and deny the program's author and publisher their rightful compensation.

How do you protect yourself against Trojan horses? As strenuously as I preach about backing up a hard disk, I must strongly warn you about Trojan horse programs:

> *Do not use any programs downloaded from public-domain bulletin boards unless you know the program has been tested first (by you or a well-known source)!*

Of the tens of thousands of public-domain and shareware programs, only twenty-five Trojan horses have been documented. But nothing is foolproof because fools are so ingenious, and those placing Trojan horses in the public domain are very great fools indeed.

Your insurance in preventing a Trojan horse from taking you for a ride is to make sure that someone has tested the program before you use it. User groups, system operators (*sysops*) of reputable bulletin boards, and associates screen programs before distributing them. These are safe sources. The screening ensures that the program is not a data "terminator."

However, files that are listed on a bulletin board as "new" or "recently uploaded" files (files the sysops have not yet tested) should always be suspect. Files that have not been tested by your friends or associates should be suspect. Files from unknown sources should be suspect. If you are the first one in your crowd to receive such a program, test the program on a *floppy disk* system if possible. If the program is an indiscriminate Trojan horse and trashes your files, only the files on the floppy disk drive are lost. Be warned that one recorded Trojan horse could distinguish between a floppy disk and hard disk and would wreak havoc only on the hard disk system. If you must try out the program on your hard disk system, make sure you can afford to lose every file on your hard disk.

In response to the Trojan horse phenomenon, some public-domain programs are now available to guard against destruction.

(One example is CHK4BOMB.COM.) These programs scan a suspicious program's code for any disk-write operations or strange embedded messages (like "Aha, gotcha!!"). Another kind of insurance is a program that temporarily "locks up" (prevents most types of writing to) your hard disk and warns of any disk accesses while the suspect program is running. (One is called DPROTECT.COM.) The programs are "nice" ideas, but you should remain cautious. None of these programs can circumvent all potentially destructive disk activity, and some fool may disguise another Trojan horse as one of these programs.

As to pirated and hacked software, don't use it and don't do it. You rob yourself and others. The software industry depends on royalties and license fees to maintain support and provide the impetus for improvements. Pirated and hacked software is theft and is a motivation for publishers to use copy-protection and not improve their software.

As for shareware programs, use them and don't abuse them. If you find the program useful, send the donation. Most shareware authors complain that few people send donations. Most authors believe that the issue is not money, just the effort of sending a check. Many shareware authors are becoming former shareware authors and publishing their products through commercial publishers. For shareware to stay alive, you are on the honor system. If you find the program useful, take the time to send that modest contribution. Too many shareware bargains go unrewarded. If we want more and better shareware bargains, we need to reward the shareware authors.

Shareware Communications Software

Having properly caveated public-domain and shareware software, I'll mention two other shareware programs that are designed for accessing electronic bulletin boards (the first was PC-Talk). Knowing the chicken-and-egg problem (the programs are on many bulletin boards, but you need communication software to get the communication software), I suggest you write directly to the firms or contact friends or local user groups for current copies.

Procomm

Datastorm Technologies
P.O. Box 1471
Columbia, MO 65205

Requested donation: $25.00 no disk
$35.00 program/documentation on disk
$50.00 program disk and printed documentation

Qmodem	John Friel III Forbin Project 4945 Colfax Avenue South Minneapolis, MN 55409 Requested donation: $35.00

Public-Domain and Shareware Resources

Almost all public-domain and shareware programs listed in this book can be found on CompuServe, The Source, local bulletin boards, and local user groups. If you cannot find a local users group to assist you in obtaining these programs, try contacting the following group for information about user groups in your area:

Capital PC User Group
4520 East-West Highway
Suite 550
Bethesda, MD 20814

Programs Mentioned in This Book

The following is a list of programs discussed in this book. Public-domain programs are designated by *(pd)*. Shareware programs are designated by *(sw)*. For public-domain programs, the author, if known, is listed. Shareware programs are listed with their authors or publishers, if known. Commercial programs are listed with their publishers.

Product	Publisher/Author
1 DIR	Bourbaki Incorporated P.O. Box 2867 Boise, ID 83701
ARC (sw)	System Enhancement Associates 21 New Street Wayne, NJ 07470
ARCA (sw)	Vern Buerg
AUTOMENU	Marshall Magee Magee Enterprises 6577 Peachtree Industrial Boulevard Norcross, GA 30092

BackTrack	Tallgrass Technologies Corporation 11100 West 82nd Street Overland Park, KS 66214
CED (pd) PCED	Chris Dunford Cove Software Group 10057-2 Windstream Drive Columbia, MD 21044
CompuServe	CompuServe Corporation
CopyII PC PC Tools	Central Point Software 9700 S. W. Capitol Highway Portland, OR 97219
DiskOptimizer	SoftLogic Solutions, Inc. 520 Chestnut Street Manchester, NH 03101
DPATH20 (pd)	James A. McGreggor, Jr.
DSBackup	Design Software Incorporated 2 N. 520 Prince Crossing Road West Chicago, IL 60185
Extended Batch Language (sw)	Seaware Corporation P.O. Box 1656 Delray Beach, FL 33444
FastBack	Fifth Generation Systems 7942 Picardy Avenue Baton Rouge, LA 70809
FilePath FilePaq	SDA Associates P.O. Box 36152 San Jose, CA 95158
Flash	Software Masters 6223 Carrollton Avenue Indianapolis, IN 46220
GO (pd)	C. DeVoney, T. Leslie, and A. Stegamoller
G-WHIZ (sw)	Gordon Waite P.O. Box 1862 Iowa City, IA 52244
Lightning	Personal Computer Support Group 11035 Harry Hines Boulevard Dallas, TX 75229

LU (pd)	T. Jennings
Mace Utilities HTEST/HFORMAT	Paul Mace Software, Inc. 123 First Street Ashland, OR 97520
NUSQ03 (pd)	Dave Rand, Paul Homchick, and Cliff Sharp
PathMinder	Westlake Data Corporation P.O. Box 1711 Austin, TX 78767
PCOPY (sw) PMOVE (sw) PSEARCH (sw)	Norm Patriquin
PC-File	Buttonware, Inc. P.O. Box 5786 Bellevue, WA 98006
PCSWEEP	Koch Software Industries 11 West College Drive Arlington Heights, IL 60004
PC-Talk (sw)	The Headlands Press P.O. Box 8962 Tiburon, CA 94920
PKARC (sw)	Phil Katz
Q&A	Symantec Corporation 10201 Torre Avenue Cupertino, CA 95014
RENDIR (pd)	
SpeedStor	Storage Dimensions 14127 Capri Drive, Suite 1 Los Gatos, CA 95030
Superkey	Borland International 4585 Scotts Valley Drive Scotts Valley, CA 95066
The Norton Utilities The Norton Commander	Peter Norton 2210 Wilshire Boulevard Santa Monica, CA 90403

VCACHE VFeaturePlus	Golden Bow Systems 2870 Fifth Avenue, Suite 201 San Diego, CA 92103
VTREE (pd)	
Watchdog	Fischer Innis Systems Corp. 4175 Merchantile Avenue Naples, FL 33942
XTREE	Executive Systems Incorporated 15300 Ventura Boulevard Sherman Oaks, CA 91403
ZAP (pd)	

Index

* Ordinarily, command syntax is found on the first page reference. An asterisk indicates where the command syntax is first presented on a later page.

E

I-K

L

M

N-O

T

More Computer Knowledge from Que

WORD-PROCESSING AND DESKTOP PUBLISHING TITLES

Using DisplayWrite	19.95
Using Microsoft Word, 2nd Edition	19.95
Using MultiMate Advantage, 2nd Edition	18.95
Using PageMaker on the IBM	24.95
Using WordPerfect, 3rd Edition	19.95
Using WordStar	18.95
WordPerfect Tips, Tricks, and Traps	19.95

DATABASE TITLES

dBASE III Plus Applications Library	19.95
dBASE III Plus Handbook, 2nd Edition	19.95
dBASE III Plus Advanced Programming, 2nd Edition	22.95
dBASE III Plus Tips, Tricks, and Traps	19.95
R:BASE System V Techniques and Applications	19.95
R:BASE System V User's Guide, 2nd Edition	19.95
Reflex Tips, Tricks, and Traps	19.95
Using Reflex	19.95
Using Paradox	21.95
Using Q & A	19.95

APPLICATIONS SOFTWARE TITLES

Excel Macro Library	19.95
Multiplan Models for Business	15.95
Smart Tips, Tricks, and Traps	23.95
Using AppleWorks, 2nd Edition	19.95
Using Dollars and Sense	16.95
Using Dollars and Sense on the IBM	17.95
Using Enable	19.95
Using Excel	19.95
Using Javelin	21.95
Using Managing Your Money	17.95
Using Smart	22.95
Using SuperCalc4	18.95

IBM TITLES

Networking IBM PCs, 2nd Edition	19.95
Using PC DOS	21.95

LOTUS SOFTWARE TITLES

1-2-3 for Business	$19.95
1-2-3 Business Formula Handbook	19.95
1-2-3 Command Language	19.95
1-2-3 Macro Library, 2nd Edition	19.95
1-2-3 Tips, Tricks, and Traps, 2nd Edition	19.95
Using 1-2-3, Special Edition	24.95
Using 1-2-3 Workbook and Disk, 2nd Edition	29.95
Using Lotus HAL	19.95
Using Symphony	23.95
Symphony: Advanced Topics	19.95
Symphony Macros and the Command Language	22.95
Symphony Tips, Tricks, and Traps	21.95

COMPUTER SYSTEMS TITLES

Amiga Programming Guide	18.95
CP/M Programmer's Encyclopedia	19.95
Managing Your Hard Disk	19.95
MS-DOS User's Guide, 2nd Edition	21.95
Using NetWare	24.95

PROGRAMMING AND TECHNICAL TITLES

Advanced C: Techniques and Applications	21.95
C Programmer's Library	21.95
C Programming Guide, 2nd Edition	19.95
C Self-Study Guide	16.95
C Standard Library	21.95
Common C Functions	18.95
Debugging C	19.95
Turbo Pascal for BASIC Programmers	16.95
Turbo Pascal Program Library	19.95
Turbo Pascal Tips, Tricks, and Traps	19.95
Understanding UNIX: A Conceptual Guide	21.95
Understanding XENIX: A Conceptual Guide	21.95
Using Turbo Prolog	19.95

que™

Que Order Line: **1-800-428-5331** All prices subject to change without notice.

MORE COMPUTER KNOWLEDGE FROM QUE

Using PC DOS
by Chris DeVoney

In the lucid, easy-to-understand style that made him a best-selling author, Chris DeVoney describes both the common and not-so-common operations of PC DOS. DeVoney guides users—both novice and intermediate—through basic and advanced DOS commands. A Command Reference defines every DOS command, gives examples, and tells how to handle common problems. *Using PC DOS* is two books in one—a concise tutorial and a valuable reference you will refer to over and over again.

dBASE III Plus Handbook, 2nd Edition
by George T. Chou, Ph.D.

A complete, easy-to-understand guide to dBASE III Plus commands and features. The *Handbook* provides full discussion of basic and advanced operations for displaying and editing data. Numerous examples of databases show the options for handling, sorting, summarizing, and indexing data. The book explains error messages in detail and offers tips on how to avoid errors when you create files and enter data. For both newcomers to dBASE III and former dBASE III users, the *dBase III Plus Handbook,* 2nd Edition, will help you manage your office or business more efficiently.

Using 1-2-3, Special Edition

Now the best book on 1-2-3 is even better! *Using 1-2-3* has helped more than one million readers get "up and running" with Lotus 1-2-3. This Special Edition adds more than 300 feature-packed pages to the highly acclaimed tutorial, including comprehensive *Troubleshooting* and *Command Reference* sections. All-new "hands-on" practice sessions help you master difficult topics, and tips and techniques placed throughout the book improve your efficiency with 1-2-3. Join the more than *one million* satisfied readers who have unleashed the power of 1-2-3 with *Using 1-2-3!*

Using WordPerfect, 3rd Edition
by Walton Beacham and Deborah Beacham

The 3rd Edition of this consistent best-seller features a totally new design and over 150 additional pages covering the many features of 4.2. Reorganized to follow the way the WordPerfect user learns this top-selling word-processing package, *Using WordPerfect,* 3rd Edition, has a progressive, new modular design to guide the user easily through the book and to serve as a handy reference book later. New "Quick Start" exercises provide immediate application of the concepts and information presented. A 4-color tear-out keyboard reference offers hands-on assistance.

Mail to: Que Corporation • P. O. Box 90 • Carmel, IN 46032

Item	Title	Price	Quantity	Extension
180	Using PC DOS	$21.95		
68	dBASE III Plus Handbook, 2nd Edition	$19.95		
805	Using 1-2-3, Special Edition	$24.95		
11	Using WordPerfect, 3rd Edition	$19.95		
			Book Subtotal	
			Shipping & Handling ($2.50 per item)	
			Indiana Residents Add 5% Sales Tax	
			GRAND TOTAL	

Method of Payment:

☐ Check ☐ VISA ☐ MasterCard ☐ American Express

Card Number _____ Exp. Date _____

Cardholder's Name _____

Ship to _____

Address _____

City _____ State _____ ZIP _____

If you can't wait, call **1-800-428-5331** and order TODAY.

All prices subject to change without notice.

MYHD-8712

FOLD HERE

Place
Stamp
Here

Que Corporation
P.O. Box 90
Carmel, IN 46032

REGISTER YOUR COPY OF *MANAGING YOUR HARD DISK*

Register your copy of *Managing Your Hard Disk* and receive information about Que's newest products. Complete this registration card and return it to Que Corporation, P.O. Box 90, Carmel, IN 46032.

Name _____

Company _____ Title _____

City _____ State _____ ZIP _____

Phone _____

Where did you buy your copy of *Managing Your Hard Disk*?

_____

How do you plan to use the information in this book?

_____

_____

_____

What other kinds of publications about computer systems would you be interested in?

_____

_____

_____

_____

Which operating system do you use? _____

THANK YOU!

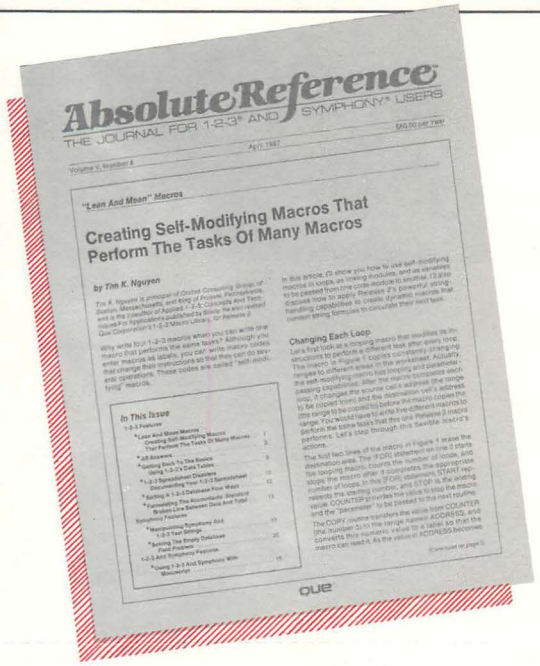
MYHD-8612

FOLD HERE

Place
Stamp
Here

Que Corporation
P.O. Box 90
Carmel, IN 46032

BECOME A MORE PRODUCTIVE 1-2-3® AND SYMPHONY® USER!


Absolute Reference
THE JOURNAL FOR 1-2-3® AND SYMPHONY® USERS

"Lean And Mean" Macros

Creating Self-Modifying Macros That Perform The Tasks Of Many Macros

by Tim K. Nguyen

In This Issue

Changing Each Loop

que

You'll increase your productivity when you apply the timesaving tips and macros in Absolute Reference, the monthly journal devoted to helping you become a Lotus power user.

Each Absolute Reference issue provides efficient business applications designed to fit your busy schedule. You also get a question-and-answer column, expert reviews, and helpful tips and traps—many of which are written by the same 1-2-3 and Symphony experts who have authored quality Que books.

Join the thousands of other Lotus users who are already applying the 1-2-3 and Symphony applications they've learned from Absolute Reference. Subscribe today and start increasing your productivity with the first issue.

FREE FIRST ISSUE

We'll process your subscription, send our latest issue FREE, and bill you for your subscription. If Absolute Reference is not for you, write "cancel" on the invoice and return it. Or, submit payment and receive 12 more issues—your first one's FREE.

que

YES, I want to become more productive. Start my subscription with my first FREE issue.

METHOD OF PAYMENT:

☐ Check Enclosed ☐ Bill me ☐ Charge Card

Card # _____ Exp. Date _____

Cardholder's name _____

☐ 12 issues plus 1 FREE for $60 ($80 outside U.S.A.)
☐ 24 issues plus 1 FREE for $105 ($135 outside U.S.A.)

Name _____ Title _____

Company _____

Address _____

City _____ State _____ ZIP _____

Country _____ Day Phone _____

MAIL THIS CARD OR CALL 1-800-227-7999 EXT. 555

Allow 6-8 weeks for delivery

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

First Class Permit No. 9918 Indianapolis, IN

Postage will be paid by addressee

Absolute Reference™

THE JOURNAL FOR 1-2-3® AND SYMPHONY® USERS

11711 N. College Avenue
Carmel, IN 46032

Another quality que publication

MANAGING YOUR HARD DISK

The definitive guide to selecting, installing, and using a hard disk, **Managing Your Hard Disk** gives you the techniques and tips you need to use your hard disk more efficiently. This book shows you how to take advantage of many DOS commands and how to avoid common pitfalls when using a hard disk. With this book, you will build a toolbox of programs that make the hard disk easier to use—and totally indispensable.

Managing Your Hard Disk is divided into four parts, each one devoted to a different aspect of using a hard disk:

Part I gives you background information on hard disks, including important points for selecting and installing the right hard disk for your computer.

Part II discusses key DOS concepts and commands for managing and controlling your hard disk and presents the most sophisticated examples of DOS batch files.

Part III is an invaluable tutorial and reference for solving hard disk problems, from the most mundane to the most complex. This section describes both common and not-so-common hard disk management problems and shows you effective techniques for solving them.

Part IV covers programs designed to simplify your hard disk use, techniques for making trouble-free hard disk backups, and valuable tips on installing many popular software packages.

Managing Your Hard Disk is written by Don Berliner with Chris DeVoney. Mr. Berliner, a 23-year veteran of the computer industry, is an internal consultant to a Fortune 200 company and author of numerous business and computer articles. Mr. DeVoney, contributor to the book, is author of several best-sellers by Que, including *Using PC DOS*.

Whether you are a beginning or experienced hard disk user, you will benefit from the insightful and easily understood advice presented in this book.

$19.95 USA

67

que™

0-88022-265-4